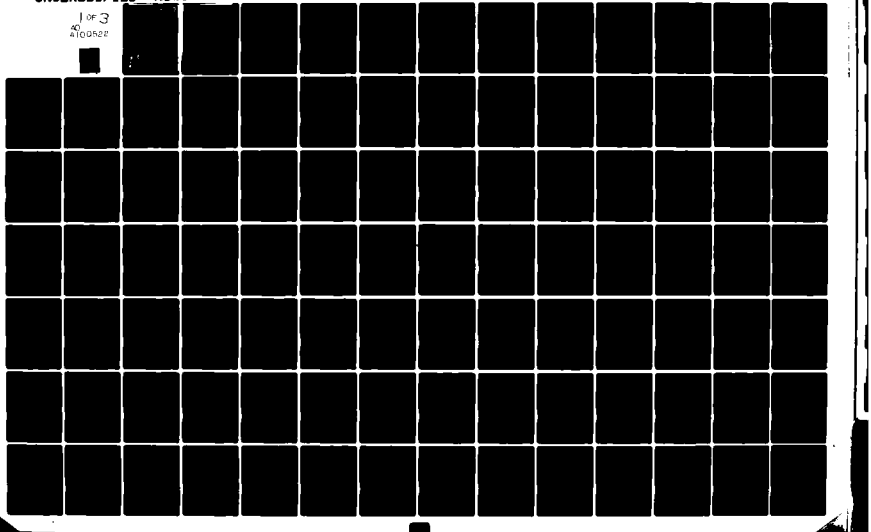
AD-A100 522 NAVAL SURFACE WEAPONS CENTER DAHLGREN VA F/G 9/2
EMULATION AID SYSTEM II (EASY II) SYSTEM PROGRAMMER'S GUIDE.(U)
MAR 81 C J NAPLES
UNCLASSIFIED NSWC/TR-81-98 NL

1 OF 3
AD A100522



NSWC TR 81-98

EMULATION AID SYSTEM II (EASY II) SYSTEM PROGRAMMER'S GUIDE

by
CHARLES J. NAPLES, Editor
Strategic Systems Department

MARCH 1981

NAVAL SURFACE WEAPONS CENTER

Dahlgren, Virginia 22448 Silver Spring, Maryland 20910

NAVAL SURFACE WEAPONS CENTER
Dahlgren, Virginia 22448

Paul L. Anderson, Capt., USN
Commander

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC TR-81-98	2. GOVT ACCESSION NO. AD-A100	3. RECIPIENT'S CATALOG NUMBER 522
4. TITLE *(and Subtitle)* EMULATION AID SYSTEM II (EASY II) SYSTEM PROGRAMMER'S GUIDE		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Charles J. Naples, Editor		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (K33) Dahlgren, VA 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NIF
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Weapons Center Dahlgren, VA 22448		12. REPORT DATE March 1981 13. NUMBER OF PAGES 208
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Virtual Machines	Target Machines	Task Control Program (TCP)
Virtual Machine Monitor	Intermediate Language Machine	
Emulation	Resource Allocation	
Microprogramming	SIMPL-Q Programming Language	
Operating Systems	Command Interpreter	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report is a system programmer's guide for the Emulation Aid System II (EASY II). EASY II is an operating system/virtual machine monitor that runs on the Nanodata QM-1 microprogrammable computer.

EASY II provides integrated support for:

1. Interactive control of multiple concurrently resident

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (Continued)

emulations each run in a stand-alone manner;

2. Interactive control of networks of emulations involving communication and synchronization among processors;

3. Virtual, as well as real input/output of emulators to various real peripherals; and

4. Diagnostic displays for debugging of both the emulations and software running on the emulated computers.

FOREWORD

This report is a system programmer's guide for the Emulation Aid System II (EASY II), an operating system/virtual machine monitor implemented on the NSWC Nanodata QM-1 microprogrammable computer. EASY II is the successor of EASY.

The system described in this report was developed in the Programming Systems Branch of the Computer Division. The EASY II project members consisted of Wayne Russell, Steven Derry, Chuck Flink (Principal Designer of EASY), Alan [illegible], Jay Meyers, Ronald Hartung, Charles Naples, John Perry, and Helen Fletcher.

This report was prepared with the assistance of Paul Tiffany, Donald Oates, and Carolyn Tilghman of System Development Corporation (SDC). Special thanks to Anne Ammerman who was the principle editor of the original EASY System Programmer's Guide.

This report was prepared in the Programming Systems Branch of the Computer Division and reviewed by Hermon W. Thombs, Acting Head, Programming Systems Branch.

Released by:

R. T. RYLAND, JR., Head
Strategic Systems Department

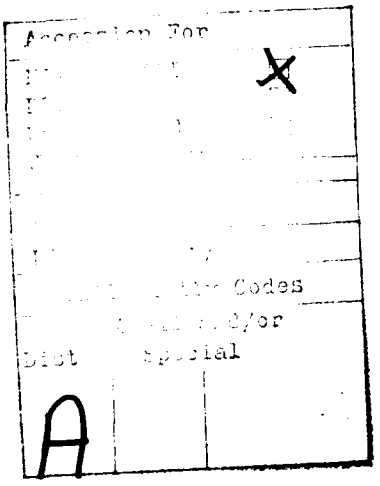


TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

I. INTRODUCTION

The Emulation Aid System Version Two (EASY-II) was designed and implemented by the Computer Division, Programming Systems Branch of the Naval Surface Weapons Center/Dahlgren Laboratory. EASY-II is the successor of EASY. The major difference between the two systems is the incorporation of a Virtual Machine Monitor (VMM) under EASY-II.

EASY-II provides integrated support for:

1. Interactive control of multiple concurrently resident emulations each run in a stand-alone manner;

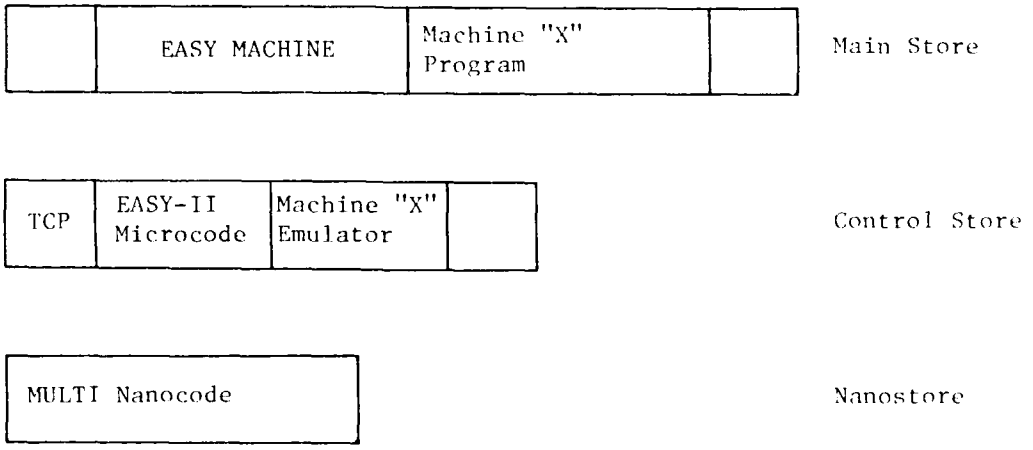
2. Interactive control of networks of emulations involving communication and synchronization among processors;

3. Virtual, as well as real input/output of emulators to various real peripherals; and

4. Diagnostic displays for debugging of both the emulations and software running on the emulated computers.

I.1 MACHINE ARCHITECTURE AND ITS RELATION TO THE MAJOR CONCEPTS OF EASY-II

EASY-II is implemented on the Nanodata QM-1 microprogrammable computer. The QM-1 has three distinct levels of memory: main store (MS), control store (CS), and nanostore (NS). Each memory level contains one or more major components of the EASY-II system.



I.1.1 Main Store

The memory that is traditionally thought of as main memory (where application programs are stored during execution) is called Main Store (MS).

In a typical EASY-II system configuration, Main Store would contain the following:

1. EASY Machines*
2. Emulated Target Machine Memory

The EASY machine is the operating system of the EASY-II system. It contains such things as a command processor, peripheral utilities, interrupt handlers, etc. The EASY machine also contains the Virtual Machine Monitor (VMM).

Elements of the EASY machine are written in a high-level, Algol-like language, SIMPL-Q. The SIMPL-Q compiler generates an intermediate language called E-CODE. The collection of E-CODE modules and its associated stack space is called an EASY machine. It is worthy to note that because EASY-II supports a Virtual Machine Monitor (VMM), multiple EASY machines may run concurrently in Main Store.

In addition to EASY machines, main store also contains emulated memory for target machines being hosted under EASY-II. In other words, emulated memory for machine "X" would contain programs used by the typical applications programmer of machine "X".

I.1.2 Control Store

The QM-1 is microprogrammable at two levels. The first level of microprogrammability is at the Control Store (CS) level. Control Store contains programs coded in the virtual microprogramming (highly encoded, 18-bit instructions) language, MULTI.

Control Store contains the following components of the EASY-II system. As indicated above, each component is coded in MULTI.

1. Task Control Program (TCP)
2. EASY-II Emulator
3. Target Machine Emulators

TCP is a kernel operating system providing low-level support in such areas as interrupt processing, scheduling, resource allocation, task switching, etc. The EASY-II emulator is essentially an interpreter of main store E-CODE.** Target machine emulators interpret the target machine application progress in Main Store and provide an environment consistent with the target machine architecture.

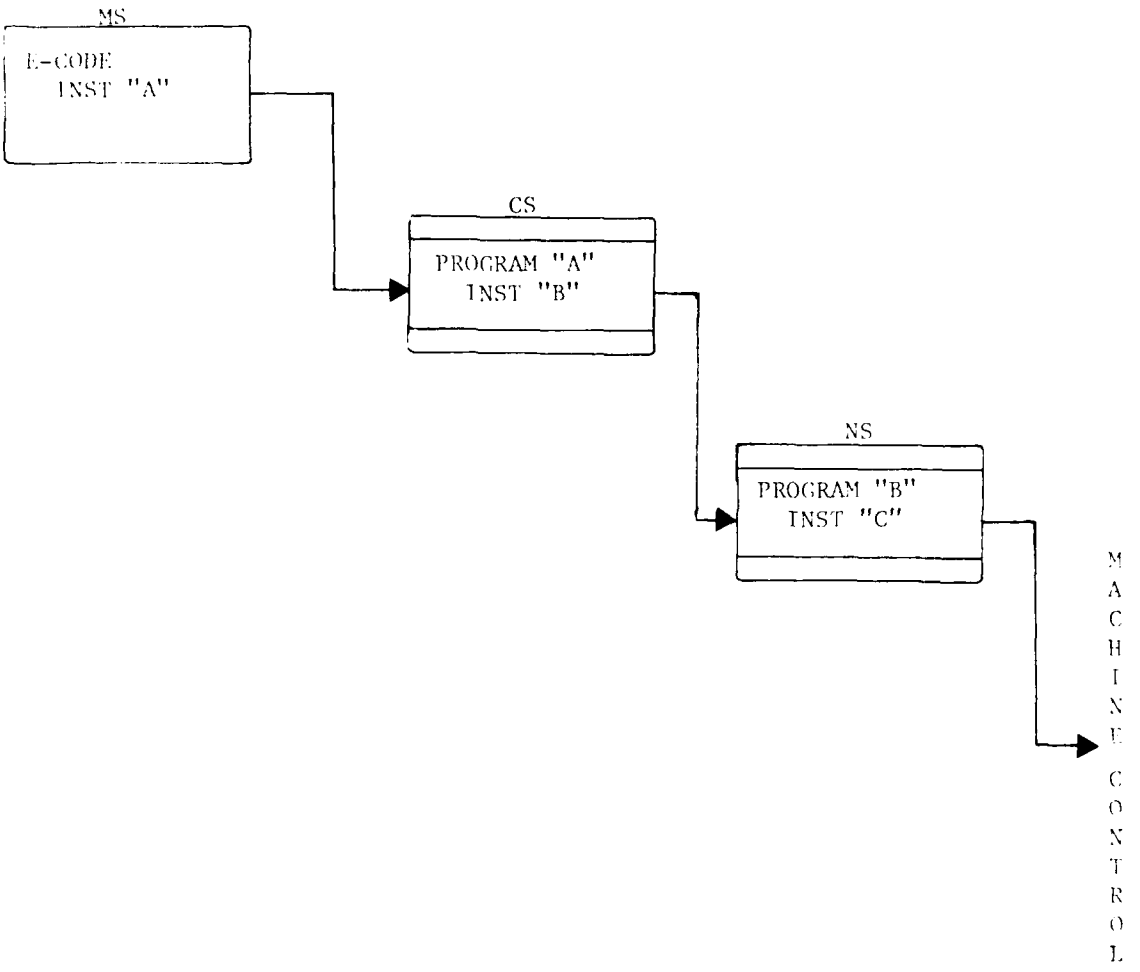
---

* Actually, EASY machines are emulated target machines.

** Actually, an intermediate language machine.

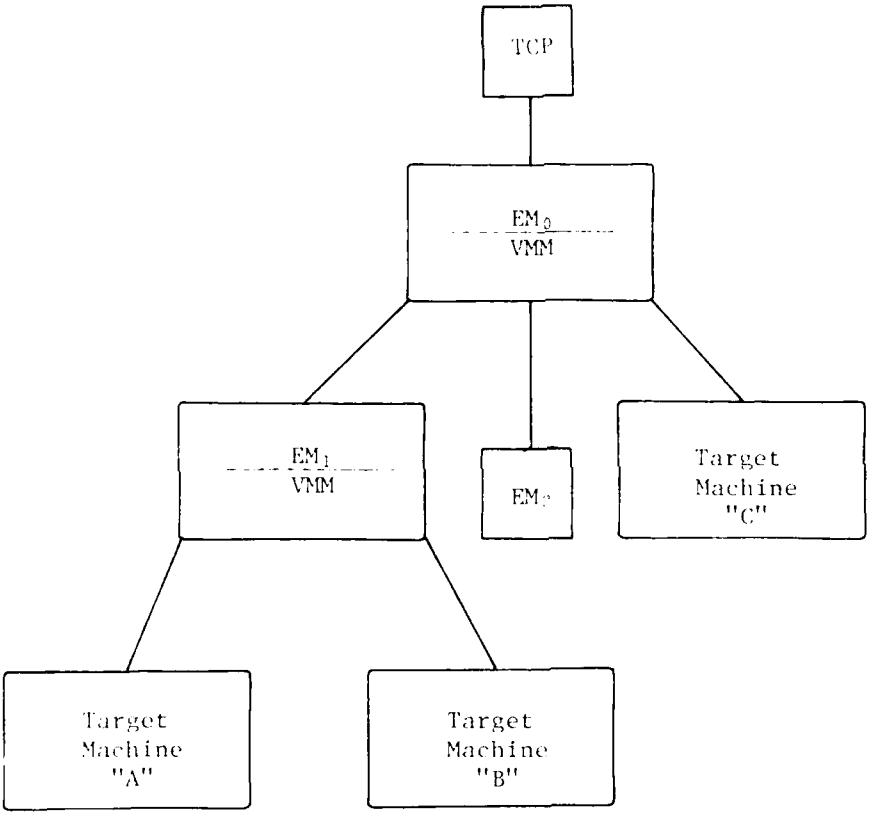
1.1.3 Nanostore

Nanostore contains horizontal microcode (minimally encoded, 360-bit instructions). That is, each instruction in Nanostore achieves a high degree of parallelism and is directly related to primitive hardware functions of the QM-1. It is the function of Nanostore programs to interpret the virtual microcode instruction in control store. Thus, a hierarchical effect is achieved between the three levels of memory.



1.1.4 Relationship Between TCP, EASY Machines, and Target Machine Emulations

TCP, EASY Machines, and Target Machine emulations exist in a parent-sibling relationship. TCP is the kernel operating system of the QM-1 and as such provides the necessary support for task (emulation) switching, resource allocation, low-level I/O, and service functions of value to emulator writers. TCP is the parent of all tasks running under EASY-II. At deadstart time, TCP allocates, out of the QM-1's resources, an EASY Machine (called EM_0). EM_0 provides the user with typical user-interface utilities (via its role as an operating system) and the ability to appropriate, out of EM_0 resources (via the Virtual Machine Monitor), additional EASY machines or Target machine emulations.



These additional EASY machines may again allocate portions of their resources to other EASY machines or Target machine emulations.

II. MASTER

II.1 GENERAL USE OF MASTER

MASTER is the command interpreter for EASY. It is a table-driven command interpreter designed to accept commands and their parameters from the console keyboard, disk, or cards; check syntax of parameters (applying defaults in place of null parameters); and, finally, pass the parameters to the appropriate routine for that command. The table defining a legal command is known as the "command template." The collection of command templates available for consideration by MASTER on any given invocation is called the "command file." The standard operating mode of MASTER is to accept input from the console keyboard. For changing the input medium of MASTER, see use of the EXEC command (Section V.19).

When "!!" appears on the CRT, it means that MASTER is awaiting a command. Each command is identified by a unique keyword. If a user desires to know what commands are available, the user must type a question mark ("?") to have a table of command keywords presented. If the user needs further information about a specific command, the user can type the command's keyword immediately followed by a question mark. In response to this, the command's template will be displayed, defining the purpose, syntax, defaults, and other information about the command. Appendix A describes the process required to bring up the EASY Emulator. As described in A.7.e, the system awaits a response of date and time to initialize the system. If the user types a question mark, he will get the CMDINI command file as shown in Figure 1. In Figure 2, the user has typed DATE followed by a question mark to obtain the DATE command template.

```
!!?
COMMANDS FOR INITIAL TIME AND DATE SET.
AVAILABLE COMMANDS: (FILE CMDINI)
<@>     DATE            TIME
**********
```

Figure 1. CMDINI Command File

```
!!DATE?
DATE,=19S79S99.    (MM/DD/YY)
SET DATE.
01/00/00.
$SDATE.
```

Figure 2. DATE Command Template

The template for a given command consists of four card images (see Figure 3). The first card defines the command syntax. The second is a comment outlining the use of this command. The third card image lists defaults for the various parameters in proper order, and the last card provides the name of the "Server" routine to be executed in response to the command.

The command syntax defined by the first card will be of the following format:

$$\text{KEYWORD,PROMPT1=}\begin{matrix}\text{choice field}\\ \text{type field}\end{matrix}\text{,...,PROMPTn=}\begin{matrix}\text{choice field}\\ \text{type field}\end{matrix}$$

The prompts allow MASTER to display to the user on the CRT what parameter is needed. The type field defines for MASTER the syntax of the expected response from the user so MASTER may do syntax checking.

The syntax field types and their meanings are:

Type	Acceptable Characters
1	"0" or "1" (binary)
7	"0" thru "7" (octal)
9	"0" thru "9" (decimal)
F	"0" thru "9" and "A" thru "F" (hex)
A	"A" thru "Z" or "$" (alpha)
N	"A" thru "Z", "0" thru "9", or "$" (alphanumeric)
S	any character (special)

Instead of using PROMPTn=type field, the user may use the choice syntax which is a series of choices each preceded by an !. The format is:

```
PROMPT=!option!option!option!
```

For example, if the user requests the EDITLIB template by typing an EDITLIB? after the !!, he will observe the following:

```
EDITLIB,LIBRARY=SSSSSSSSSS,STATUS=!OLD!NEW!.
EDIT SYSTEM LIBRARIES
USERLIB,OLD.
EDTLIB.
```

Command keywords can be up to 14 characters in length. As soon as the user has typed in sufficient characters for MASTER to detect uniqueness, MASTER will automatically complete the command keyword for the user. For example, in the CPX command file, keywords which start with M are:

MONITOR-PRINT, MASTER, MONITOR-RESTA, MONITOR-DUMP,
MONITOR-OFF, and MONITOR-ON

To indicate the MONITOR-OFF keyword, the operator types MOOF. When MO is typed, the system will show MONITOR-. By typing OF after the dash, the system will then automatically pick up the final F. In this way descriptive keywords can be used without requiring excessive typing on the part of the user. After

MASTER has filled in the rest of the keyword, the user must acknowledge the keyword by typing a comma or a space. MASTER then starts prompting for parameters using the prompts from that command's template. When typing the keyword, the operator does not need to wait for the system to fill in the rest of the word. If the operator types excess characters, a beep will sound indicating invalid entry. Care should be taken when a beep occurs to verify that the keyword entry matches the desired entry. For example, if MONITOR was typed from the CPX command file, the MO would generate MONITOR-, the NIT would generate three beeps, the O would result in a MONITOR-O, and the R would generate a beep.

Using Figure 3 as an example, after the user has typed in "COPY" followed by a space or comma, MASTER will display the prompt "FROM=" and await a response from the user of up to 10 characters of any kind. The user types in the parameter and terminates it with a comma or space. MASTER then prompts for the next parameters and so on until all the parameters have been entered, whereupon MASTER executes the command. If the user does not type in anything for the parameter (i.e., replies with only space or comma), the default, as defined on the 3rd card of the command template, will be taken. If the user tries to enter anything that is syntactically incorrect, the CRT will beep. Examples are: an illegal keyword, too many characters in a parameter, a character that is unacceptable according to the syntax type field specification of a parameter.

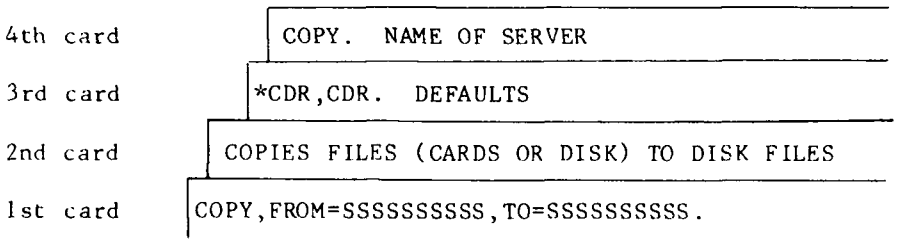


Figure 3. Sample Template, COPY Command

MASTER recognizes several special keys. They are:

Key	Meaning
Space or Comma	Terminate this keyword (or parameter) and go on to the next one. Execute command if this is last parameter.
Period or Carriage Return	Terminate this keyword (or parameter). Select defaults for all remaining parameters (if any), and execute the command.
Backspace or Rubout	Backspaces within this parameter or keyword.
Control-X or Control-U	Disregard the entire input line and start over.

Key	Meaning
Control-Z, Control-C or Control-D	Abort the EASY system and reinitialize it. NOTE: These keys work at all times.
Question Mark in place of a keyword	Produce list of all available keywords.
Question Mark immediately following a keyword	Display command template.

MASTER is capable of recognizing certain other one-key commands. Some of these are used by subsystems of EASY. When they are used, the writeup will explain their use. To determine if a certain key has a meaning at a particular point, type its complete symbolic name (exactly as shown below) followed by a question mark. Its template will be displayed if it is a valid command at that point.

Key	Complete Symbolic Name
Carriage Return	<CR>
Period	<PERIOD>
Line Feed	<LF>
Escape	<ESC>
Control-K	<CONTROL-K>
Asterisk	<*>
Plus	<+>
Minus	<->
"@"	<@>

Certain EASY commands, when executed, bring up a different command set. Normally, the "@" command will return the user to the original set of commands. From the CMDINI command file, @ will place the user at the system command file, CMDSYS.

II.2 PRIVILEGED COMMANDS

The EASY system is always operating in one of two command modes - privileged or unprivileged. When running in unprivileged or LOCKED mode, the user is not allowed access to certain commands (i.e., MASTER will not recognize the privileged commands if typed). In order to operate in privileged or UNLOCKED mode, the user must know the system password. Those commands available only in privileged mode are those which, if used carelessly, have the potential for causing havoc in the system. The initial state of EASY is LOCKED. The commands LOCK and UNLOCK are used to switch back and forth between privileged and unprivileged mode. The description of a command will indicate it is privileged. The command template for UNLOCK is as follows:

```
UNLOCK,PASSWORD=SSSSSSSSSS.
UNLOCKS KEYWORD.
ESP.
UNLOCK.
```

Now that the EASY system is developed, the lock/unlock capability is rarely used. LOCK is an example of a locked command. Thus, once the user types LOCK, the EASY system will no longer recognize the lock entry until UNLOCK is used.

II.3 USE OF MASTER IN SYSTEM DEVELOPMENT

The use of MASTER from the point of view of an EASY (SIMPL-Q) user seated at the console has already been described (i.e., how it responds to user inputs, etc.). MASTER was designed, first of all, to be easily used by the novice. A second goal in the design of MASTER was to make it easy to use by the system developer. Heavy prompting, parameter checking, and default management can consume a great deal of the time and energy of the system designer. If MASTER is used for the greatest part of the "user interface" portion of a system, then this design problem for that portion reduces to the design of the command templates and layout of the command files.

Since MASTER is reentrant, it may be called recursively (i.e., it may be called from a server running under a higher level of MASTER). Thus, the system designer would take the following steps to have the system (coded in SIMPL-Q) incorporated into the EASY system:

1. Create command templates for the new system and corresponding server routines.

2. Create a command file of these templates using the utility MAKECF. For example, MAKECF, INPUT=S:CMDYK20,COMMANDFILE=CMDYK20.

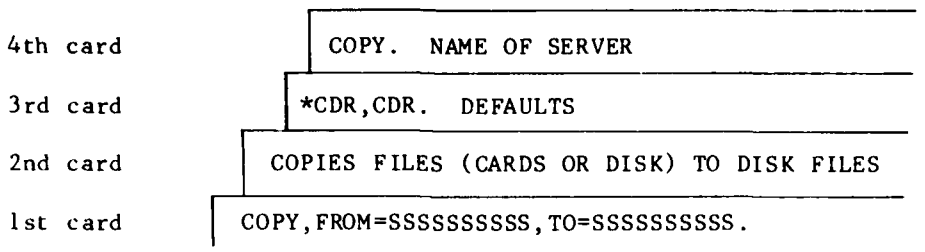
3. Create a server routine to call MASTER with this new command file. For example, YK20XF is the UYK-20 server. It is referenced under CPX (see UYK-20CPEF command template). CPX is a command of the EASY system.

4. Create a command template for a command to invoke the new system (may be name of system) and ask NSWC to add this template to the appropriate EASY command file such as CMDSYS or CMDCPX. The 4th card of this template should indicate the server routine that will call MASTER with a new command file.

Step 4 should be taken only when the system has been checked out and is ready for user use. The system should be checked out using the EASY server TEST (see Section V.49).

II.4 COMMAND TEMPLATE FORMAT

The template for a given command consists of four card images. The first card defines the command syntax. The second is a comment describing the use of this command. The third card image lists defaults for the various parameters in proper order, and the last card provides the name of the server routine to be executed in response to this command.



The command syntax defined by the first card must be of the following format:

$$\text{KEYWORD,PROMPT1=}\begin{vmatrix}\text{choice field}\\ \text{type field}\end{vmatrix}\text{,...,PROMPTn=}\begin{vmatrix}\text{choice field}\\ \text{type field.}\end{vmatrix}$$

The sequence must begin with the keyword identifying the command, optionally followed by up to eight parameter fields. The keyword and each parameter field must be followed by a comma, except for the last, which must be terminated by a period. The keyword may consist of any characters except blanks, "!", commas, question marks, or periods. It may be no longer than 13 characters if the command is not locked, and no longer than 14 characters if the command is locked. A parameter field consists of a user prompt and a syntax type specification or a set of choices for the response. The prompt will be displayed by MASTER to the user. Each prompt must be followed by an "=" and may consist of any characters except commas, periods, or "=". The syntax type fields follow to define the type of response acceptable in answer to the corresponding prompt. The syntax character types are:

Type	Acceptable Characters
1	"0" or "1" (binary)
7	"0" thru "7" (octal)
9	"0" thru "9" (decimal)
F	"0" thru "9" and "A" thru "F" (hex)
A	"A" thru "Z" or "$" (alpha)
N	"A" thru "Z", "0" thru "9", or "$" (alphanumeric)
S	any character (special)

The syntax type fields may contain only the syntax characters listed above and be no longer than 20 characters. The field must specify the maximum number of characters to be allowed as well as the type of the character to be accepted. This is done in the following manner: If a maximum of n characters of syntax type x are to be allowed, then the syntax type field has the form:

$$x\ x\ \ldots x_n$$

Syntax types may be mixed in a field as desired.

Where the choice syntax is used, each choice is preceded by a ! and the final choice ends with a !. There is no limit on the number of choices, but each choice must not contain more than 20 characters. An example is the command EDITLIB.

EDITLIB,LIBRARY=SSSSSSSSS,STATUS=!OLD!NEW!.

The syntax of the third card is:

DEFAULT1,DEFAULT2,...DEFAULTn.

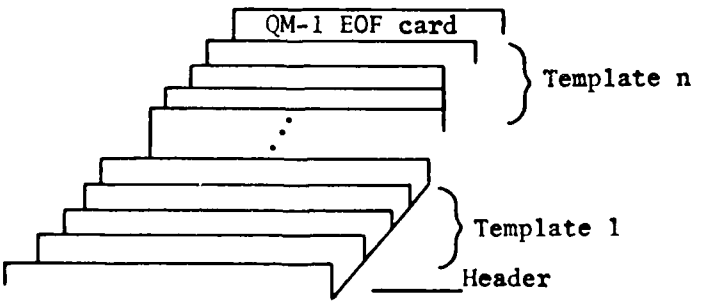
This card must contain a default for each parameter in the command, and they must be separated by commas and terminated by a period. The defaults must not contain commas or periods and cannot be longer than 20 characters. If the user does not type in a response to a prompt, the default is passed for the corresponding parameter. The default is not checked against the syntax, so defaults may be programmed which the user cannot specify.

The fourth card gives the name of the "server" routine to be executed, followed by a period. It must be a valid entry name. If the use of the command is to be restricted to only those knowing the system password, then the name of the routine should be followed by ",LOCKED.".

Following the period, any card in the template may contain a comment.

II.5 COMMAND FILE FORMAT

The command file is the collection of command templates which define the commands currently acceptable to MASTER. A command file is built from a command deck by the utility MAKECF. A command deck is made up of a Header card followed by n sets of four-card command templates followed by a QM-1 EOF card.



The Header is a single card which is a comment describing the system of commands the command file is set up to support. There must be a four-card template for each command. The keyword for each command in a command file must be unique. The command file is a SIMPL-Q random disk file. The keywords will be arranged in alphabetical order and prefixed by an exclamation point (!) by MAKECF.

11.6 ONE-KEY IMMEDIATE COMMANDS

MASTER has the capability of recognizing the following one-key commands. In order to take advantage of this, the programmer must provide a template in the command file for each key to be used.

KEY	Keyword on Template
Carriage Return	<CR>
Period	<PERIOD>
Line Feed	<LF>
Escape	<ESC>
Control-K	<CONTROL-K>
Asterisk	<*>
Plus	<+>
Minus	<->
"@"	<@>

The spelling of the keyword on the template must be exactly as shown above including the "<" and ">" symbols. Within the EASY system certain commands, when executed, bring up a different command set (command file). Normally, the "@" key has been used to mean return to previous level.

11.7 RELATIONSHIP BETWEEN MASTER AND A SUBSYSTEM

11.7.1 Calling MASTER

MASTER must first be called to establish a new level of operation. MASTER has a single argument, a string. The string is taken to be the name of the command file this level of MASTER will operate under. MASTER will return to the calling routine when one of its servers executes a RETURN.

11.7.2 Linkage between MASTER and a Server

Any routine to be executed as a server under MASTER must be declared as an ENTRY PROC with one argument - a string array. The parameters for the command are passed via the string array, with the first parameter in element 0, the second in element 1, etc. These parameters have already been checked against the specified syntax and defaults have been applied.

A server may terminate in one of four ways:

1. A normal return to MASTER - which causes MASTER to prompt for a keyword from the same command file. This is effected by executing a RESTART(MASTER). See the SIMPL-Q Reference Manual for use of the intrinsic routine RESTART.

2. A termination return - which causes MASTER to return to its original caller (may be a server running under a higher level of MASTER). This is done by executing a RETURN.

3. An abort with a message to the CRT. MASTER will then prompt for a command from the same file. This should be called, for instance, when a server has encountered an error in the user input. This abort can be effected by a CALL ABORTM('error msg').

4. A fatal abort - which will flash a blinking error message on the CRT and produce a traceback. The system will then be dropped and must be restarted from scratch. A fatal abort will be produced by a CALL ABORTF('error msg'). ABORTF should only be called when an error condition which should not exist is detected and which will probably require a systems programmer's attention. Since ABORTF drops the running system, it should not be called indiscriminately.

The externals MASTER, ABORTM, and ABORTF are declared in SYSLIB.SYSET [i.e., must have !INCLUDE(SYSLIB.SYSET), a macro pass command, in compilation].

II.7.3 Standard Servers Available

MASTER provides five standard servers which may be called from any command file. They are:

1. EXEC - Executes a file of commands.

2. LOCKBD - Locks keyboard.

3. UNLOCK - Unlocks keyboard if correct password has been supplied.

4. LEAVE - Executes a RETURN (for the <@> command).

5. NOTI - Displays the message "COMMAND NOT YET IMPLEMENTED" on the CRT.

III. SIMPL-Q SUPPORT

III.1 INTRODUCTION

This chapter provides a general description of SIMPL-Q support facilities and is intended as a guide to the systems programmer. Subjects discussed include: SYSLIB (the system linkage file), SYSMAC (the system macro file), intrinsics, and libraries.

III.2 SYSLIB

SYSLIB is the SIMPL-Q system source file which provides linkage to commonly used SIMPL-Q routines. The file is comprised of three types of COMDECKS (common decks): SETUP, CODE, and GLOBALS.

SETUP COMDECKS provide linkage to commonly used procedures via the external declaration feature (EXT) of SIMPL-Q. The following is a list of SETUP COMDECKS.

Name	Description
SYSET	Linkage to EASY system routines
CONSET	Linkage to EASY conversion routines
FPSET	Linkage to floating point routines
MTSET	Linkage to magnetic tape handlers
KBDSET	Linkage to keyboard handlers
CRTSET	Linkage to CRT handlers
CIOSET	Linkage to disk handlers
DNTSET	Linkage to Dynamic Debug for TRIDENT (DDT) file handlers

CODE and GLOBAL COMDECKS are SIMPL-Q source decks which may be included in the programmer's source code. The following is a list of CODE and GLOBAL COMDECK.

Name	Type	Description
SYSTXT	CODE	Contains source code for systems routines
TRDGLB	GLOBALS	Globals for TRIDENT emulation
PHDEFS	GLOBALS	Path handler definitions
MASTDEF	GLOBALS	Configuration constants for MASTER

III.2.1 SYSET - Linkage to EASY System Routines

SYSET contains the linkage to commonly used routines. These procedures and functions are divided functionally into the following categories: general, control card analyzation, and program space manipulation.

General

The following routines are contained in EASY, core-resident, and may be linked to by !INCLUDE(SYSLIB.SYSET).

CALL ABORTM(STRING MSG)

This routine puts an error message up on the CRT and returns control to MASTER. This routine should be called by any server which detects a nonfatal error condition.

CALL ABORTF(STRING MSG)

This routine flashes an error message on the CRT and aborts. It should be called by a server only when it detects a fatal error; i.e., a condition which should not exist, etc.

CALL ATTACH(FILE F, STRING FILENAME)

ATTACH will place the file name into the FET (File Environment Table) for F. Any action on F should be complete before calling ATTACH and the name should adhere to NOVA conventions (Appendix H).

STRING FUNC SYSHDR

SYSHDR will return an 80-character string. The first 40 characters are the command last executed by MASTER. The remaining 40 are the system id, time, and date.

CHAR FUNC UPPERCASE(CHAR)

UPPERCASE will return the upper case equivalent ASCII character.

CALL MASTIN(STRING)

MASTIN sets the execute file name for the next MASTER command input. A null string implies keyboard input. "*CDR" indicates card reader input. " FILENAME " is used to indicate disk file input.

Control Card Analyzation Package

The following routines are contained in EASY, core-resident, and may be linked to by !INCLUDE(SYSLIB.SYSET).

These subroutines are an aid to the programmer who analyzes control cards, etc. They provide a standardized word scanning system as follows.

All words are written in a single string, separated by commas, and terminated by a period. They may be of any length and may contain any characters other than commas and periods.

Examples - Word1, Word2, Word3. 3 Words
Word1. NOTE: Anything after . is ignored, there are no words on this line (no period).

The following subroutines are provided:

```
INT FUNC WORDCNT(STRING CARD)
```

Returns number of words on card.

```
INT FUNC SFIND(STRING LOOK, STRING CARD)
```

Returns a 1 if LOOK is a word on CARD, 0 otherwise.

```
INT FUNC OCCURRENCES(STRING, STRING)
```

Returns number of occurrences of 2nd string in 1st.

```
STRING FUNC REPLICATE(STRING, INT)
```

Returns a string of integer replications of a string.

```
STRING FUNC SWORD(INT I,STRING CARD)
```

Returns the Ith word on CARD, or a null string if there is no Ith word on the card.

```
STRING FUNC SDATE
```

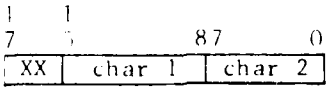
Returns current date as 'MM/DD/YY'.

```
STRING FUNC STIME
```

Returns current time as 'HH:MM:SS'.

```
CALL READCF(FILE,REF STRING, REF INT)
```

READCF provides the capability to read NOVA format coded files. The specified file is read sequentially, and the record (up to 80 characters) is placed into the string. The integer is set to false if there was valid data, true if an EOI was encountered. The format is as follows:



Two 8-bit ASCII characters are packed into one SIMPL-Q character (18 bits) with two high-order don't care bits. Linefeeds and nulls are ignored, and a carriage return terminates each record.

Program Space Manipulation Routine

The following routines are contained in EASY, core-resident, and may be linked to by !INCLUDE('YS'I','YSLT).

These routines allow for the marking and releasing of program space. Prior to the addition of these routines, the intrinsics PSFREE and PSLOAD were the only available routines for manipulating program space. The problem was that modules might be inadvertently removed by PSFREE which the user wished to maintain in

program space. With the addition of PSMARK and PSRELEASE, the user can lock modules into program space and free modules under program control.

The following routines are provided:

```
CALL PSMARK(REF INT)
```

Marks the program space stack and references mark with a marker integer.

```
CALL PSRELEASE(REF INT)
```

Releases the allocated structures to a previously marked location. An example is as follows:

```
PSMARK(I)                    Mark Program Space
ALLOCATE(INT ARRAY,N)  |     Allocate
ALLOCATE(CHR ARRAY,M)  |       Data Structures
   .
   .
   .
PSRELEASE(I)                 Release Structures
```

III.2.2 CONSET-SIMPL-Q Conversion Routines

There are seven SIMPL-Q conversion routines which may be linked to via the system library COMDECK CONSET; i.e.,

```
!INCLUDE(SYSLIB.CONSET)
```

```
STRING FUNC $EXIST(INT)
```

$EXTST breaks the 36-bit integer parameter into six 6-bit fields. Each field is viewed as a 6-bit ASCII representation of a character in the set (A-Z0-9$). $EXTST returns a 6-character string whose characters correspond to the six 6-bit fields of the input parameter.

```
INT FUNC $STEXT(STRING)
```

$STEXT takes the first six characters of the input string, converts them into six 6-bit ASCII fields, and packs them into a 36-bit integer which is returned to the caller.

```
CALL $STPCK(REF STRING, INT, INT ARRAY)
```

$STPCK packs the left-most specified number of characters (parameter 2) from REF STRING into an integer array (four char/wrd). The format is as follows.

```
XXAAAAAAAAABBBBBBBBBXXCCCCCCCCCDDDDDDDD
```

where X = don't care bit
A = first character, etc.

INT FUNC ASCII4(STRING NAME)

Returns up to four ASCII characters as nine bits left-justified, zero-filled integer.

```
Examples - FET(1): = ASCII4(NAME)
           FET(2): = ASCII4(NAME[4])
           FET(3): = ASCII4(NAME[9])
```

INT FUNC SEXTCK(STRING)

Returns a 1 if string is a valid external name. Otherwise, it returns a zero.

STRING FUNC OCT12$(INT)

Returns a string of length 12 (leading zeros) which contains the base 8-character string equivalent of the integer argument.

STRING FUNC OCT20$(INT)

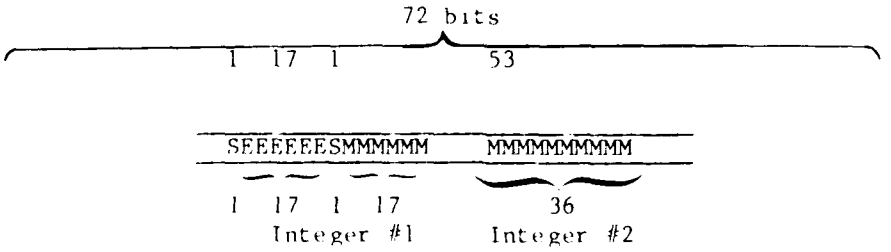
Returns a string of length 20 (leading zeros) which contains the base 8-character string equivalent of the integer argument.

III.2.3 FPSET - LINKAGE TO THE FLOATING EASY SOFTWARE PACKAGE

The floating EASY package contains SIMPL-Q routines to perform the basic floating point (F.P.) operations. The EASY F.P. format uses two SIMPL-Q integers (72 bits) to represent a number. Besides the basic F.P. operations, several conversion routines are provided. These routines may be linked to by !INCLUDE(SYSLIB.FPSET).

III.2.3.1 Floating Point Format

Two SIMPL-Q integers (72 bits) are used to represent floating point numbers.



Exponent is two's complement (18 bits). Mantissa is signed magnitude (54 bits). Floating point zero = 0 _____ 0. All F.P. numbers must be normalized.

```
Examples -  1.0 = (Ø'1225',0)        .1   x 2
           10.0 = (Ø'4242'0)         .1010  x 2
```

NOTE: Zn implies followed by n zeros.

III.2.3.2 Floating Point Arithmetic

The four basic arithmetic functions add, subtract, multiply, and divide are provided. The routines and their parameters are described in the table below. Note that the "I" means integer and "RI" means a reference integer. Note also, since EASY F.P. requires two words (integer), each operand of the arithmetic operation is a composite of two separate integers.

Name	Type	No. of Parms.	Parms.	Function
$FPADD	PROC	6	I,I,I,I,RI,RI	Addition
$FPSUB	PROC	6	I,I,I,I,RI,RI	Subtraction
$FPMUL	PROC	6	I,I,I,I,RI,RI	Multiplication
$FPDIV	PROC	6	I,I,I,I,RI,RI	Division

```
Examples - INT A1, A2, B1, B2, C1, C2
           CALL $FPADD(A1,A2,B1,B2,C1,C2)     /*C:=A+B*/
           CALL $FPSUB(A1,A2,B1,B2,C1,C2)     /*C:=A-B*/
           CALL $FPMUL(A1,A2,B1,B2,C1,C2)     /*C:=A*B*/
           CALL $FPDIV(A1,A2,B1,B2,C1,C2)     /*C:=A/B*/
```

III.2.3.3 Floating Point Conversion Routines

In addition to F.P. arithmetic operations, the floating EASY package has several conversion routines to convert floating point to and from a SIMPL-Q string and to and from an integer. Below is a list of these conversion routines and examples of their use. Note that each F.P. number requires two integer parameters.

Name	Type	No. of Parms.	Parms.	Function
$FPCBD	STRINGFUNC*	2	I,I	F.P. to STRING
$FPCDB	PROC	4	S**,RI,RI,RI	STRING to F.P. The 2nd Parm. is the error code return (0 if no error or 1 if error).

* STRING FUNC: Always returns a string of length 25 in this format.
' ±.1234567890123456789012345 E±1234'

** STRING FORMAT FOR A FLOATING NUMBER IS:
(b) (±) (d) (.) (d) (b) (E) (b) (±) (d) (b)

Name	Type	No. of Parms.	Parms.	Function
$FPCIE	PROC	3	I,RI,RI	Integer to F.P.
$FPCEI	INT FUNC	2	I,I	F.P. to Integer

Examples - S:=$FPCBD(A1,A2)
CALL $FPCDB(S,ERROR,R1,R2)
CALL $FPCIE(I,R1,R2)
I:=$FPCEI(A1,A2)

11.2.3.4 TRIDENT Base Processor F.P. Support

The floating EASY package includes the following conversion routines to convert from TBP F.P. to EASY F.P. and from EASY F.P. to TBP F.P.

Name	Type	No. of Parms.	Parms.	Function
$FPCET	PROC	4	I,I,RI,RI	EASY to TBP
$FPCTE	PROC	4	I,I,RI,RI	TBP to EASY

Examples - CALL $FPCET(A1,A2,R1,R2)
CALL $FPCTE(A1,A2,R1,R2)

11.2.4 I/O ROUTINES

11.2.4.1 MTSET - Linkage to Magnetic Tape Routines

The magnetic tape I/O routines $MT and $MTR provide the user with the necessary tape reading and writing facilities. These routines must be linked to via INCLUDE(SYSLIB.MTSET).

CALL $MT(P1,P2,P3,P4,P5,P6)
CALL $MTR(P1,P2,P3,P4,P5,P6,P7)

P1 - Tape drive - either 0 or 1

P2 - Integer array buffer for $MT and character array buffer for $MTR into/from which data is transferred (dummy array for operations where no data transfer occurs).

P3 - Number of QM-1 words transferred (dummy value for operations where no data transfer occurs).

P4 - Tape operation:
$MTREAD - Read
$MTWRITE - Write
$MTEOF - Write tape mark
$MTSKIPF - Search file forward
$MTBKSP - Backspace
$MTUNLD - Unload
$MTREWD - Rewind
$MTSTATUS - Status request

P5 - EOF/EOT flag 1 means EOF/EOT encountered
                   0 means EOF/EOT not encountered

P6 - Parity designator - 0 odd parity
                         1 even parity

P7 - Residual Word Count - requested word count minus the number of words actually transferred to buffer

III.2.4.2 KBDSET - Keyboard Linkage to SIMPL-Q Read Routines

The Keyboard Input Routine $KBCHAR is a string function which returns one character in string format from the CRT keyboard. This routine must be linked to by !INCLUDE(SYSLIB.KBDSET).

```
Linkage:  EXT STRING FUNC $KBCHAR
Sample Usage:  S:= S .CON. $KBCHAR
```

$KBCHAR may be used successively to assimilate a line of keyboard input. It is up to the user to determine when a line of input has been completed by recognition of some character which the user has denoted as the end of line, such as carriage return or line feed, etc. Additional keyboard input routines are listed below.

```
EXT PROC $KBINIT              - EASY deadstart initializer
EXT PROC KBDREAD(REF STRING)  - Reads a string in and echos the string to the
                                CRT
EXT PROC INTREAD(REF INT)     - Reads an integer in and echos the integer to
                                the CRT
```

III.2.4.3 CRTSET - Linkage to CRT Routines

Three routines are provided to support the CRT. They may be linked to by !INCLUDE(SYSLIB.CRTSET).

```
$THERE(X coordinate, Y coordinate)
$SCRTØ(string S)
$WHERE(X coordinate, Y coordinate)
```

$THERE is a string function which returns a string consisting of a set cursor directive, an x coordinate, and a y coordinate. This string is suitable to be passed on to the CRT output routine $SCRTØ. The x, y coordinates input must be an integer.

$SCRTØ is a procedure which handles writing to the CRT. The input string parameter may contain any of the following: Field Attribute Codes, cursor positioning directives, output messages, control directions (ring bell, etc.). $PCRTØ was designed to replace $SCRTØ as a means of obtaining hard copy. This routine is not currently part of CRTSET and has not replaced $SCRTØ. Sophisticated displays using cursor positioning, ruling characters, and reverse video features can be handled by $PCRTO as long as the following conditions are satisfied. The most esoteric displays (e.g., those that blank the screen and are built from bottom to top or backwards) are not handled by the utility.

1) All ruling characters are converted to a character specified at compile time (typically '*').

2) Displays are assumed to roll onto the CRT from the bottom of the screen (the y-coordinate on cursor positioning commands is ignored). Each line must terminate with the string $CRLF (as supplied by SYSLIB). Line fragments will be ignored.

3) The parameter string must not have any one line that expands to more than 120 characters, though any number of lines may be passed.

4) Ruling characters (14 & 15) must be paired on each line using this feature.

5) Cursor positioning directives are padded from the current position within a line to the x-coordinate (should never be > 79). Always progress from left to right.

6) 'US', FAC pairs are replaced by a space.

7) Any second parameter besides 'ON' will suppress the hard copy and go only to the CRT.

8) Special characters (e.g., 'HT', 'VT', 'BEL', etc.) will go directly to the printer as whatever translation is associated with the respective display code and will occupy 1 character position.

There are two parameters for $PCRTO which are described below. The following declaration is required to use this utility.

```
EXT PROC $PCRTO(STRING, STRING)
```

Parameters:

1) String to go to 4023 CRT and optionally to line printer. This string should meet the criteria outlined above.

2) 'ON' will cause a hard copy to be generated. Any other parameter will inhibit hard copy.

$WHERE is a procedure which returns the current (X,Y) position of the cursor in integer form.

NOTE: The X coordinate must be in the Range 0-79.
The Y coordinate must be in the Range 0-23.
The (0,0) position is at the top left-hand corner of the screen.

The following table lists the Field Attribute Codes (FAC) which specify how data is to be displayed. Each of these codes is a constant string of length 2.

Example linkage: EXT ASCII STRING $WØB1

Table 1. Field Attribute Codes (FAC)

Logic Effect / Display Effect	Transmittable				Non-Transmittable			
	Unprotected		Protected		Unprotected		Protected	
	alpha-numeric	non-alpha	normal	dim	alpha-numeric	non-alpha	normal	dim
White on Black	$WØB1	$WØB2	$WØB3	$WØB4	$WØB5	$WØB6	$WØB7	$WØB8
Black on White	$BØW1	$BØW2	$BØW3	$BØW4	$BØW5	$BØW6	$BØW7	$BØW8
Blinking	$BNK1	$BNK2	$BNK3	$BNK4	$BNK5	$BNK6	$BNK7	$BNK8
Blanked	$BLK1	$BLK2	$BLK3	$BLK4	$BLK5	$BLK6	$BLK7	$BLK8

The following list gives the controls available. Each of these is a constant string and must be declared EXT ASCII STRING in the program using them. The length of the string is given in brackets.

```
$BEL[1]  /*Ring Bell */
$BS[1]   /*Back Space */
$HT[1]   /*Tab */
$LF[1]   /*Line Feed */
$VT[1]   /*Back Tab */
$CR[1]   /*Carriage Return */
$EFF[2]  /*Clear Page*/
$ERC[2]  /*Read Cursor */
$NUL[4]  /*Null Fill */
$CRLF[6] /*Carriage Return, Line Feed */
$FS[1]   /*Set Cursor Directive */
```

Example of CRT Routine Use

```
EXT STRING FUNC $THERE(INT,INT)
EXT PROC $SCRTØ(STRING)
EXT PROC $WHERE(INT,INT)
EXT ASCII STRING $CR, $LF, $NUL, $BLNK1

CALL $WHERE(XSAVE,YSAVE)
S:= $CR .CØN. $LF .CØN. $NUL .CØN.
    $THERE(XPUT,YPUT) .CØN. $BLNK1.CØN. 'message'
CALL $SCRTØ(S)
CALL $SCRTØ($THERE(XSAVE,YSAVE))
```

III.2.4.4 CIOSET - Linkage to $CIO

$CIO is the driver routine for all of CIO. CIO is the general-purpose disk I/O package designed to support sequential and random access file I/O. It is described in more detail in Appendix I of this document.

```
PROC $CIO(INT,INT ARRAY,INT)
```

The first parameter is the request code. The integer array gives the FET address and the final parameter is the recall type.

III.2.5 ENTSET - Linkage to Dynamic Debug for TRIDENT (DDT) File Handlers

ENTSET provides the linkage to DDT file handlers. The file manipulation described by the following routines provide dynamic file allocation and deallocation within the context of DDT. Actually, dynamic DDT files are really data base type entries within a large, static QM-1 file. The QM-1 system does not provide dynamic file manipulation. Thus, dynamic file properties could only be realized through a data base under program control.

The procedures available are FHOPEN, FHLIST, FHCRE8E, FHUPDTE, FHDLETE, FHFINDE, LSTENTRY, and FHCLOSE. Linkage is also provided to string function FHNTHSTR. This section provides a brief description of these procedures and the associated parameters.

```
CALL FHOPEN(FILE)
```

This procedure opens the file.

```
CALL FHLIST(FILE,FILE)
```

This procedure lists all entries on the file.

```
CALL FHCRE8E(FILE,STRING,REF INT,REF INT)
```

This creates an entry on the specified file. String is the entry name and the first reference integer is the index. The final parameter is the error flag. A value of zero implies no error. A full description of error codes follows the procedure descriptions.

CALL FHUPDTE(FILE,INT,STRING,STRING,STRING,REF INT)

This procedure is used to add, delete, or replace a string in a file. The integer is the entry index. Parameter 3, the first string, indicates A, D, or R for add, delete, or replace. Parameter 4 is the old string and parameter 5 is the string which replaces it. Parameter 6 is the error code which is explained following the procedure descriptions.

CALL FHDLETE(FILE,STRING,REF INT)

This procedure deletes an entry. The string is the entry name and the integer is the error code which is explained following the procedure descriptions.

CALL FHFINDE(FILE,STRING,REF INT,REF INT,REF INT)

This procedure finds the named entry. Integer parameters 3-5 represent the index, number of records, and the error code. The error code explanation follows the procedure descriptions.

CALL LSTENTRY(FILE,INT,FILE)

This procedure gets all entry information off the file specified by parameter 1 with an index of parameter 2. Parameter 3 is the scratch file on which the entry records are placed.

CALL FHNTHSTR(FILE,INT,INT)

This is a string function which gets the nth string of the entry. Parameter two is the index and parameter 3 is the record number.

CALL FHCLOSE(FILE)

This procedure closes a file.

The meaning of the error codes in FHCRE8E, FHUPDTE, FHDLETE, and FHFINDE is as follows:

0 - No error
1 - Entry not in directory
2 - Entry already in directory
3 - Insufficient space
4 - Illegal string
5 - Unrecognized request

III.3 SYSMAC

SYSMAC is a source file of MACROS. SYSMAC consists of documentation MACROS which are on COMDECK, DOCMAC, and 7 definition COMDECKS, DEF1, DEF2, DEF3, DEF4, DEF5, DEF6, and DEF7. The definitions are included in the compile when the definitions are required by the function being compiled. A brief description of each COMDECK follows.

COMDECK DOCMAC consists of various documentation macros which are used for internal program documentation. The use of SIMPL MACROS is described in Appendix VII of the SIMPL-Q Reference Manual. The documentation macros are made available to the SIMPL-Q programmer by placing the !INCLUDE(SYSMAC.DOCMAC) at the beginning of each compile unit. The major macros of DOCMAC are !MODULE, !PARAMETER, !REVISION, !ERROR, !CONTINUE, and !FINISH. These are described in detail in Appendix X of the SIMPL-Q Reference Manual. !MODULE is used to provide an overall description of the compile unit. !PARAMETER describes parameters. !REVISION documents program changes including programmer, date, and reason for change. !ERROR is used to describe error codes. !CONTINUE extends the other macros beyond 40 characters. !FINISH indicates the end of the documentation MACROS.

DEF1 - System Functions and ECB Assignments

This COMDECK consists of system functions and Event Control Block (ECB) assignments. The system functions are parameters to the intrinsic. Appendix K lists the system intrinsics giving a brief description of each one. The ECB assignments are for the keyboard, CIO, disk, clock, system (error trap), shared ECB for CRT, line printer and card reader, tape drive, TBP, DCP, and control Q.

DEF2 - Circular I/O

This COMDECK defines the global constants for File Name Tables (FNTs) and File Environment Tables (FETs). Appendix I presents a summary of CIO including FET and FNT descriptions.

DEF3 - System Error Codes

COMDECK DEF3 defines the system error codes. These errors are described in Appendix B following the EASY emulator error codes. The error codes described include sequential I/O, printer, random access I/O, word addressable I/O, CRT, card reader, CIO, and miscellaneous errors.

DEF4 - CIO Codes

COMDECK DEF4 contains the CIO request codes as described in Appendix I.

DEF5 - Miscellaneous MACROS

COMDECK DEF5 is reserved for miscellaneous macros. It currently contains the macro for Mod 2**N and the tape, printer, and reader IDs for obtaining I/O device types.

DEF6 - DCP/TBP Macros

COMDECK DEF6 contains system recall definitions, and TRIDENT Basic Processor (TBP) definitions, and Data Communications Processor (DCP).

DEF7 - Logical Device IDs for TCP

COMDECK DEF7 contains the logical device IDs for the Task Control Program (TCP) including IDS for the CRT, card reader, line printer, magnetic tape, disk drives, communications controller, and two auxiliary CRTs.

III.4 INTRINSICS

Intrinsics are routines available to the programmer in which the system provides linkage. There are three types of intrinsics available to the SIMPL-Q programmer. These are: intrinsics implemented by the compiler or through the compiler by calls to other SIMPL-Q routines; function type intrinsics which are mapped to microcode programs; and the SYSTEM* intrinsics in which the first parameter defines the desired function. All three intrinsic types are listed in Appendix K of this document and Appendix VI of the SIMPL-Q Reference Manual.

III.5 LIBRARIES

Libraries contain SIMPL-Q object modules. When the system cannot find a reference (procedure or data item) in memory, it will search specified libraries in a predetermined order in an attempt to resolve the reference. There are two main libraries used by the system. They are SYSTEM and SIMPLQLIB.

SYSTEM contains all EASY operating system routines and is available at deadstart. SIMPLQLIB contains all routines necessary to run the SIMPL-Q compiler. EASY program libraries can be manipulated by using the EDITLIB command. The structure of libraries and the use of EDITLIB are described in Section V.17 of this document.

Listed below is a typical scenario which might be used in modifying operating system routines on the library SYSTEM.

1. Compile the new routines placing them on some file SSSSSSSSSS.

2. Use EDITLIB with a status of NEW to BUILD a new library. INCLUDE the new routines from the compile file SSSSSSSSSS.

3. Use the command 'LIBRARY' to specify the order of search.

```
LIBRARY,SEARCH 1ST=NEW, 2ND=SYSTEM
```

4. Once checkout is complete, use the EDITLIB command. Use the command 'REPLACE' to replace the old modules with the updates and the command 'ADD' to add any new modules.

When developing entirely new logic where the user wishes to check out the routine prior to building the command templates, the TEST command can be used (see V.49). This allows execution of a program which is not part of the EASY system library by specifying the entry point.

---

* SYSTEM is also implemented via mirocode; however, function is parameterized.

IV. CPX

IV.1 INTRODUCTION TO CPX

The primary goal of the EASY-II system is to provide control and interface mechanisms over all current and future emulators supported by TCP II on the Nanodata QM-1 computer. In most instances, some SIMPL-Q software is required which is tailored for a specific emulation. These specialized user-interface programs have been termed as control programs or CPs. CPs planned or developed include those for the CDC 200 UT emulator, the Pascal P-machine emulator, the Mk 148 (PTCC) emulator, the AN/UYK-7 emulator, the AN/UYK-20 emulator, and EASY's Intermediate Language Machine (ILM).

A drawback of these specialized CPs is the fact that CPs are developed to support single target emulators or "non-communicating" multiple emulations. This chapter describes the features of a "generalized" control program which has been termed as "CPX." CPX is designed to provide an environment in which experiments with the emulations of various network configurations (consisting of asynchronously communicating processors) may be efficiently run.

IV.2 FUNCTIONS OF CPX

The major functions of CPX are the following:

1. Control and monitor up to seven concurrently executing emulations or "processors."

2. Provide mechanisms for the allocation of a target processor's memory.

3. Support of virtual devices (e.g., printer, consoles).

4. Allocation of real devices (e.g., tape, printer).

5. Support for a "virtual time" clock to maintain limited synchronization among selected emulations.

6. Provide mechanisms to support interprocess communications.

7. Provide mechanisms to support some rudimentary forms of memory protection among all running processors.

8. Display state information about all processors as well as status of interprocessor communications.

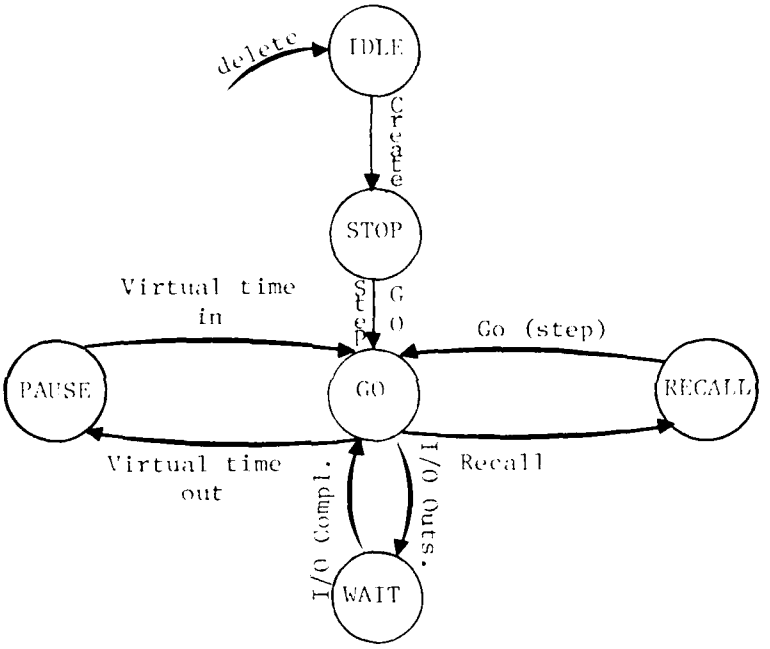
9. Provide mechanisms to save and restore the processors with the associated communications configuration.

10. Provide mechanisms to support user display interfaces for target processors.

CPX may be thought of as a "virtual machine monitor" providing resource allocation, protection mechanisms, and interprocessor communications for up to seven virtual machines running on the Nanodata QM-1. These virtual machines may be thought of as complete CPU emulations or as emulations of asynchronous elements making up a CPU. Further, the virtual machines could be a mixture of simulations and emulations running in a loosely or tightly coupled synchronized network.

IV.3 HOW CPX VIEWS A PROCESSOR

In order to better understand the user/CPX and CPX/emulator interfaces, it is necessary to illustrate how CPX sees a processor. The following state-transition diagram shows this.



The next section will describe the CPX commands as they are envisioned at the present time and how each state may be entered or exited.

The mechanism by which interprocess communications is supported is through the use of the Path Handler. This is described in Section IV.4 of this chapter.

IV.4 USER/CPX INTERFACE

CPX operates as a subsystem of the EASY-II system. The subsystem may be entered by merely typing "CPX". At this point, a number of commands become available to configure, monitor, and control an "emulated multi-processor" environment.

The following commands have been developed to be as general-purpose as possible. To avoid having the system user spending unnecessary time typing in commands, profiles of commands may be executed for experiments which are run repeatedly. The CPX commands and a short explanation of each are now presented.

1. CREATE-MEMORY,NAME= <memory name> ,WORD-SIZE=<no. of bits> , LENGTH: <no. of words> - Make a memory for the target machine, length is based on target machine's word size.

2. DELETE-MEMORY,NAME= <memory name> - If a memory is not being used by any process, then remove it.

3. CREATE-PROCESSOR,PROCESSOR= <processor number> , MEMORY= <memory name> , MACHINE TYPE= <machine type> - Make a virtual processor (go from "idle" to "stop" state). MEMORY (memory name) must be previously defined (see item 1 above).

4. GO,PROCESSORS= <list of processor numbers> - Place processor in a "go" (i.e., running) state, clearing step flag, call emulator interface routine "clear-step", only if not in either "pause" (for virtual time to complete) or "wait" (for a device) state.

5. STEP,PROCESSORS= <list of processor numbers> - If step flag is off, turn it on, call emulator interface routine "set-step"; if in "stop" or "recall" state, change state to "go" and run in a stepped manner (with stepping meaning something specific to each given emulator).

6. FREEZE,PROCESSORS= <list of processor numbers> - If freeze flag is off, turn it on; if in "go", "pause", or "wait" then halt the processor.

7. THAW,PROCESSORS= <list of processor numbers> - If freeze flag is on, turn it off; if in "go", "pause", or "wait" then start the processor.

8. SYNCHRONIZE,PROCESSORS= <list of processor numbers> or <null>, INTERVAL = <no. of time units> - Run each processor in the list for the given number of time units (1 time unit = 100 micro sec); as each processor uses up its available units, the processor goes into "pause" state until everyone in the list finishes.

9. DELETE-PROCESSOR,PROCESSORS= <list of processor numbers> - Remove the processor, placing it into an "idle" state memories (i.e., decrement user count for mainstore memories), clear buffers and release any devices assigned.

10. ASSIGN,DEVICE= <real device name>, TO-PROCESS= <processor number>, AS LDID= <logical device id> - If device is available and son is active, then give the son the real device. The devices can be a console (CON2, CON3) card reader (RDR), printer (PRTR), tape device (TAPE), clock (CLCK), line printer (LINE), or disk (DISK).

11. RELEASE-DEVICE= <real device name> - If device is assigned, then release it.

12. CREATE-PATH,PATH= <list of paths>, PATH-TYPE= <legal path types> - If path is undefined, create a path of a given type and free all subchannels. There are three path types: peripheral, intercomputer, UYK-7. Path type UYK-7 is used for paths connecting the UYK-7 CPU and the IOC which requires four subchannels (EF,OD,EIE, and ID).

13. CONNECT,PATH= <path number>,TO-PROCESSOR= <processor number>, AS-LDID= <logical device id> - If path is defined and process is active, then manipulate tables and connect path. In the case of a CP which is an emulated computer, LDID may be considered synonomous with channel.

14. DISCONNECT,PATH= <path number>,FROM-PROCESSOR= <processor number>, with-LDID - If path is defined and processor is active, then manipulate tables and disconnect path from processor.

15. DELETE-PATH,PATH= <path number>, - If path is defined, then for each processor path is connected to, make the path undefined.

16. VIRTUAL-ASSIGN,DEVICE= <virtual device name>, TO PROC= <processor number>, AS LID= <logical device id>, SPOOL= <virtual spool file name> - If son is active, assign a virtual device (e.g, printer) to receive any I/O, with I/O going to/from spool file.

17. VIRTUAL-RELEASE,DEVICE= <virtual device name>, FROM-PROCESSOR = processor number> - If virtual device is assigned, then release it.

18. FLOAT-CONSOLE,TO-PROCESSOR= <processor number> - If son is active, assign main console to him.

19. DISPLAY-STATE - Display all processors, state of processor, machine name, memories, etc.

20. DISPLAY-MEMORY - Display memory name, field length (target), and word size.

21. LDID-MAP, OF PROCESSOR=7 - For one processor, each LDID connection. following is an example of a processor with three LDIDs connected.

[illegible]	TYPE	LDID	TYPE	LDID	TYPE	LDID	TYPE
00	PATH(01)	20		40		60	
01	REAL(CON2)	21		41		61	
02		22		42		62	
03		23		43		63	
04		24		44		64	
05		25		45		65	
06		26		46		66	
07		27		47		67	
10	REAL(DISK)	30		50		70	
11		31		51		71	
12		32		52		72	
13		33		53		73	
14		34		54		74	
15		35		55		75	
16		36		56		76	
17		37		57		77	

22. DISPLAY-PATH - Display all paths with path type, subchannel state, index number of processor active on each subpath and processors connected to path, or display a single path, by specifying the path number. This displays the absolute address of the TIOCB on all active subchannels, the relative TIOCB address, control store type, and channel and transfer widths. The following example shows the path display where 'ALL' paths were requested.

```
*DISPLAY-PATHS,PATH NUMBER=ALL.
                          **********************
                          *  CPX PATH DISPLAY  *
                          **********************
   PATH        PATH      SUBCHANNEL      SONS ACTIVE        SONS
  NUMBER       TYPE        STATES      ON SUBCHANNEL     CONNECTED
  ------       ----      0123 4567       0123 4567       ---------
     1           P        ---- ----       ---- ----          14
     2           I        -R-- ----       ---- ----          13
    20           7        R--- ----       ---- ----          23
    21           7        R--- ----       ---- ----          23
    22           7        R--- ----       ---- ----          23
    23           7        RR-- W---       ---- ----          23
*FOR OLD PATH DISPLAY TYPE *DISPLAY-OLDPT*
```

The second example shows the subpaths for path 2. This indicates that son 3 has a READ outstanding to son 1.

```
                    **********************
                    *  CPX PATH DISPLAY  *
                    **********************
PATH NUMBER=02, OF PATH TYPE=INTERCOMPUTER, WITH SONS CONNECTED = 13
```

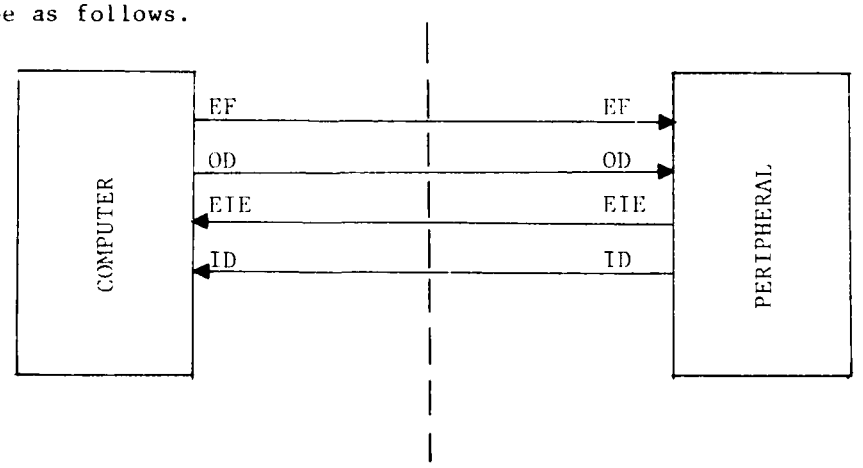
SUBPATH NUMBER	SUBPATH STATE	SON ACTIVE	TIOCB ADDRESS (ABSOLUTE)	TIOCB ADDRESS (RELATIVE)	CONTROL STORE TYPE	CHANNEL WIDTH	TRANSFER WIDTH
0	-	-	------	------	-	---	---
1	R	3	005640	000640	A	32	32
2	-	-	------	------	-	---	---
3	-	-	------	------	-	---	---
4	-	-	------	------	-	---	---
5	-	-	------	------	-	---	---
6	-	-	------	------	-	---	---
7	-	-	------	------	-	---	---

SUBPATH DISPLAY

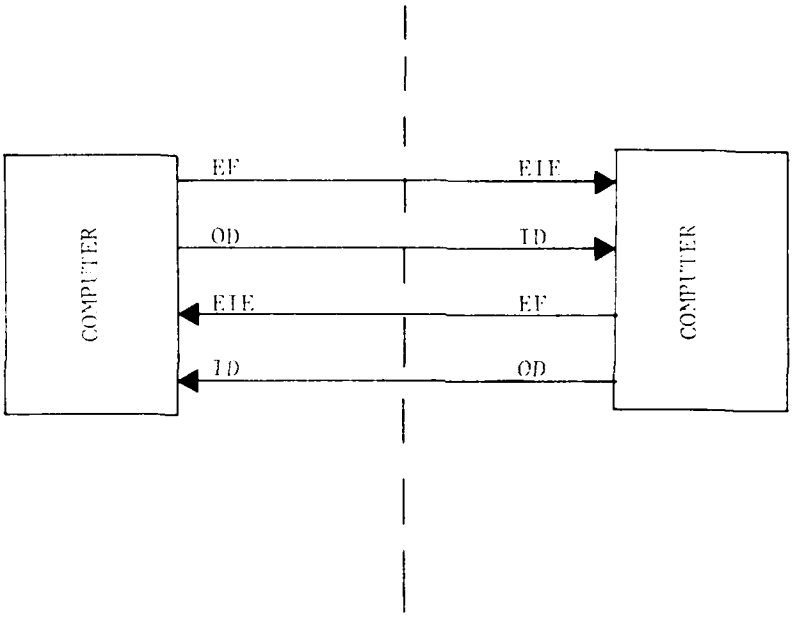
For NTDS channels, each simulated NTDS channel will correspond one to one to a path under CPX. A path is defined as a linkage, controlled by a path handler, between two or more tasks executing under EASY-II. The various functions performed by an NTDS channel will be modeled by four subchannels associated with the CPX path. The following information defines these four subchannels, their relation to actual NTDS functions, and typically how they are used in VNTDS.

Function	Subchannel
EF (External Function)	0
OD (Output Data)	1
EIE (External Interrupt Enable)	2
ID (Input Data)	3

Normally, the connection of subchannels for computer/peripheral communication will be as follows.



For intercomputer mode, the connection of subchannels is shown in the following diagram:



CPX must be made aware that an intercomputer channel mode is desired, so that the subchannels may be remapped accordingly. Data is transferred by read and write operations on the OD and ID subchannels in the indicated directions.

External functions (and external interrupts) will be sent on the EF and EIE channels. The read, write, readf, and writef constructs will be used to transfer the data. Readf and writef are used to simulate the EF with force or EI with force operations. These operations will properly simulate channel hang-ups that will occur in the real hardware.

23. DISPLAY-OLDPT - Display paths in 'OLD' format presents some of the information as displayed by DISPLAY-PATHS. This display is self-explanatory. It has been replaced in usefulness by the DISPLAY-PATHS display.

```
      0       1       2       3       4       5       6       7
PATH 01234567012345670123456701234567012345670123456701234567
TYPE  PI              7777
SUBCHO                RR R
   1   R                 R
   2  R
   3  R
   4
   5
   6
   7
```

24. DISPLAY-DEVICE - Display all real devices and who owns them.

```
                    ****************************
                    *  REAL DEVICE ALLOCATIONS *
                    ****************************
```

DEVICE NAME	PARENT TASK LDID	SON TASK LDID	PROCESSOR NO. OF SON	PROCESSOR NAME OF SON
CON2	13	01	4	EASY
CON3	11			
RDR	02			
PRTR	04			
TAPE	05			
CLCK	07			
LINE	12			
DISK	10			

25. CHECKPOINT,TO=<filename>, message=<user message> - If the processors are in proper state (FROZEN) for checkpoint, all real I/O will be halted, the disk file attached and opened, and the appropriate data will be written onto the file. This will consist of a header, CPX tables, mainstore segments, control-store-A and control-store-B segments, path tables, task tree, and associated information. The user message will be displayed upon restarting from this file.

26. RESTART,FROM=<filename> - All real I/O will be halted and all active sons idled. All mainstore segments will be released. The file specified will be attached and opened, and the information saved by CHECKPOINT will be read in. All sons will be reinitialized, and the disk file will be closed. The sons will be in the FROZEN state.

27. DEVICE-MAP,PROC=<processor number>, ARRAY=<array name>, VTOR=<virtual device number to real device number> - Map virtual to real devices. This command would be useful for devices with multiple device numbers, such as tape units or disk drives. The assumption here is that the correlation between real and virtual devices is kept in an array, the first entry representing read device 0, the second representing device 1, and so on. If the

real and virtual were initially the same but the user wished to change real devices 0 through 3 to represent virtual devices 1 through 4, he could perform the following:

```
DEVICE-MAP,PROC=4,ARRAY=IBUFF,VTOR=1234
```

The user must know the name of the array where this correlation is kept for the particular device emulator.

28. BUFFER-MAP,PROC=<processor number>, ' RAY=<array name>, SIZE=<new buffer size> - This command would normally be used to increase the size of a dynamically allocated buffer. The size of the buffer is dictated by the contents of the array named. (Note: This is not the array name of the buffer being increased.)

29. EFIOC7,PICK-IT=<processor number> - This command invokes the AN/UYK-7 IOC display interface. For a list of commands, enter <?>. These commands are described in Appendix L.

30. EFUYK7,PICK-IT=<processor number> - This command invokes the AN/UYK-7 Central Processor Unit display interface. For a list of commands, enter <?>. These commands are described in Appendix L.

31. EXEC,FILE=<filename> - This command causes a file to be executed. The file must consist of a list of commands. The following is an example of such a file which may be executed at this (CPX) level. Note that the commands are completely spelled out.

```
CREATE-MEMORY,20MEM,18,40000.
CREATE-PROCESS,1,20MEM,CP20.
CREATE-PATH,1,PERIPHERAL.
CREATE-PATH,2,INTERCOMPUTER.
CONNECT,2,1,07.
CONNECT,1,1,17.
CREATE-MEMORY,UYK7MEM,32,61000.
CREATE-PROCESS,2,UYK7MEM,UYK7.
CREATE-PATH,20,UYK-7.
CREATE-PATH,21,UYK-7.
CREATE-PATH,22,UYK-7.
CREATE-PATH,23,UYK-7.
CONNECT,20,2,20.
CONNECT,21,2,21.
CONNECT,22,2,22.
CONNECT,23,2,23.
VIRTUAL-ASSIGN,SYNC,2,77.
CREATE-PROCESS,3,UYK7MEM,IOC7.
CONNECT,2,3,3.
CONNECT,20,3,20.
CONNECT,21,3,21.
CONNECT,22,3,22.
CONNECT,23,3,23.
```

```
VIRTUAL-ASSIGN,SYNC,3,77.
CREATE-MEMORY,QRU,18,100000.
CREATE-PROCESS,4,QRU,EASY.
DIRECTORIES,'7,03,00.
LIBRARIES,QRULIB.
<CR>
ASSIGN,DISK,4,10.
ASSIGN,CON2,4,01.
CONNECT,1,4,0.
UYK20CPEF.
TAPE-TO-MEMORY.
MOD-P,1000.
<@>
EFUYK7,2.
TAPE-TO-MEMORY,1.
MASTER-CLEAR.
MODIFY-CMR,121,52400.
MODIFY-CMR,124,41150.
MODIFY-P,1,07.
<@>
<SPACE>
```

32. FLOAT-CONSOLE,TO PROCESSOR=<processor number>, CONTROL CHARS GO TO=<son or parent> - Console 1 is loaned to the specified processor which must have a virtual console. Console entries to that processor are then made through console 1.

33. UYK20CPEF,SON=<processor number> - This command invokes the AN/UYK-20 display interface which interfaces with both the AN/UYK-20 emulated IOC and CPU. For a list of commands, enter <?>. These additional commands are described in Appendix M.

34. PRINT-SPOOL,FILE=<filename> - File used as print-spool for virtual printer will be printed.

35. MASTER,COMMAND FILE=<filename> - Invoke MASTER, with command file specified, as a subsystem of CPX. An example of this command's usefulness would be a situation where a lengthy configuration has been established and the user wishes to checkpoint it, but he has forgotten the checkpoint file's name. He uses this command with the default (EASY level) command file, then uses FILES and REPORT to list the file names. He spies the file name he has forgotten, enters an <@> to return from the subsystem and his configuration is intact. Now he may proceed with his checkpointing.

The following MONITOR commands deal with recording the activity of CPX paths. The information recorded includes the following:

a. Total time any subchannel of a path is active - indexed by path number.

b. Total time each subchannel was on - indexed by path number and subchannel number.

c. Number of occurrences of each trouble code for each son - indexed by trouble codes and son number.

d. Number of occurrences of each trouble code for each path - indexed by trouble code and path number.

e. Number of occurrences of each I/O type - indexed by I/O code.

f. Number of occurrences of successful I/O - indexed by son number and path number.

g. A histogram of the number of words transferred - indexed by path number and upper limit of the bin.

36. MONITOR-ON,DEBUG PRINT=A - Turns the monitor on. Sets the debug print flag to ON if A was specified as Y (yes), or to OFF if A was specified as N (no).

37. MONITOR-OFF,FILE=<filename> - Turns the monitor off and dumps the statistics to the specified file.

38. MONITOR-RESTART-FILE=<filename>,DEBUG PRINT=A - If the monitor is off, it is restarted with the statistics from the specified file. If the monitor is on, no action is taken. In both cases, the debug print file is set to ON if A was specified as Y (yes), or to OFF if A was specified as N (no).

39. MONITOR-DUMP,FILE=<filename> - Dumps the current statistics to the specified file. This does not turn the monitor off or affect the statistics in any way.

40. MONITOR-PRINT,FILE=<filename> - Prints the statistics. If monitor is on, filename should not be entered and the current statistics will be printed. If the monitor is off, statistics in the specified file will be printed.

V. SERVERS

The routines which handle the commands are known in the EASY system as servers. Many of these are actually subsystems. As discussed earlier, MASTER, the command interpreter, is recursive. A number of these servers will call MASTER themselves with a command file of the commands they support (e.g., BIND, DISK-SAVE, EDITLIB).

In the descriptions of the syntax for the commands in this section, the user response fields will be underlined. Refer to Section II.1 or II.4 on command templates for definition of the syntax type characters used in the user response fields.

Servers	Locked
BACKSPACE	No
BIND	No
CALCULATE	No
CIMPORT	No

Servers	Locked
COMPARE	No
COMPRESS	No
COPY	No
COPYD	No
COPYNS	No
COPYSN	No
DATE	No
DDT	No
DEADSTART	No
DIRECTORY	No
DISK-SAVE/RESTORE-DISK	No
EDIT	No
EDITLIB	No
EDITOR	No
EXEC	No
FH-UTILITIES	Yes
FILES	No
IMPORT	No
LIBRARY	No
LISTCF	No
LOCK	Yes
MACRO	No
MAKECF	No
MAKERESIDENT	No
MODULATE	No
MT-TO-DISK	No
PASCAL	No
PATCH-CALC	No
PATH HANDLER LANGUAGE (PHL)	No
PRINT	No
PRINT-SPOOL	No
PRU-MOD	Yes
PRUDMP	No
QCONTROL	Yes
RDTAPE	No
REWIND	No
SIMPLQ	No
SKIPBACKWARD	No
SKIPFORWARD	No
SM-TO-DISK	No
SOURCE-TO-DISK	No
SPY	No
SPY-PROC	No
TERMSIM	No
TEST	No
TESTCF	No
TIME	No
TRANS	No
UNLOAD	No

Servers	Locked
UNLOCK	No
UYK-7	No
200UT	No
< @ >	No

V.1 BACKSPACE

This command backspaces records on a tape. End-of-File (EOF) marks count as one record when backspacing. The syntax of this command is:

BACKSPACE,UNIT=17,NUM OF RECORDS=999,PARTY=1.

where

UNIT - tape unit where tape is mounted. Default is 0.

NUM OF RECORDS - number of records to be backspaced. Default is 1.

PARITY - 0 (even), 1 (odd)

V.2 BIND

BIND is an interactive link editing subsystem for EASY program modules. The basic intent is to collect several modules and make them into one functional module.

BIND uses MASTER to interpret its commands, and is built around a word-addressable scratch file called "SCR."

The syntax for this command is:

BIND.

BIND supports the following commands:

INCLUDE	- includes modules into scratch file
RESOLVE	- resolves satisfied externals
DELETE	- deletes entry names
WRITE	- writes scratch file
MT-TO-DISK	- load SIMPLQ binary from tape
TERMINATE	- terminates BIND

INCLUDE

Syntax: INCLUDE,FILE=SSSSSSSSSS,# MODULES=9999.

Defaults: QM1,9999.

This command adds modules to the scratch file. The specified number of modules is read from the specified file. The default for "number of modules"

parameter is all of compiler output. The modules are read in order, the first modules residing in the lower addresses. Subsequent INCLUDEs start where the previous INCLUDE left off. Checks are made for valid module headers and end markers. The name and starting address of each module included is displayed on the CRT. After each INCLUDE, the length of all the modules put together is placed in the length field of the first module, thus making all the modules one composite module.

RESOLVE

Syntax: RESOLVE.

The RESOLVE command scans down the chain of externals in the composite module, permanently resolving those which are satisfied by entries in the composite module. An external is permanently resolved by filling in the link in the external marker and removing the external from the chain of external markers. All the INCLUDEs should be done before doing a RESOLVE.

DELETE

Syntax: DELETE,ENTRY=ANNNNN.

Default: $$$$$$ (Do not use the default.)

This command will delete the specified entry from the chain of entry markers. Entry name and other information will be left intact for traceback and other purposes. NOTE: Only the first occurrence of the entry name (if there are more than one) will be deleted, scanning down from the top of the composite module. This command is useful for deleting entries which should only be resolved within the composite module - in this case, perform a RESOLVE first!

WRITE

Syntax: WRITE,FILE=SSSSSSSSSS.

Default: QM2.

This command writes the current contents of the scratch file to the named file. (Without this command, BIND would be rather useless.) If a RESOLVE has not been done since the last INCLUDE, then one is performed automatically prior to writing the file.

MT-TO-DISK

Syntax: MT-TO-DISK,UNIT=1,FILE #=9999,TO FILE=SSSSSSSSSS,REWIND=S.

Defaults: 0,0,QM1, .

This command copies the binary output produced by the SIMPL-Q compiler on the CDC 6700 from tape to disk on the QM1. The UNIT parameter specifies the tape unit number, either 0 or 1 in current configuration. FILE # specifies the number of files to space forward before copying begins. TO FILE specifies the disk file name. REWIND allows the user three options, B (rewind before file searching begins), A (rewind after copy), and blank (no rewind). Tape records must be

1,536 frames long with six good bits per frame. (This is format of 6700 SIMPL-Q compiler output.) Tape records will be reformatted into 256 SIMPL-Q word (i.e., 36 bits) blocks and written sequentially to disk.

TERMINATE

Syntax: <@>. (MAY BE IMMEDIATELY EXECUTED BY A @.)

This command will terminate the operation of BIND, but it will do so only if a WRITE has been done since the scratch file was last changed (by an INCLUDE or DELETE). If the file has not been saved, then a @ will cause a warning to be issued and control will remain in BIND. A second @, however, will allow the user to leave BIND without saving the composite module.

V.3 CALCULATE

The CALCULATE command allows the user to perform computations and to use the QM-1 as a calculator while working at the console. The syntax of this command is:

CALCULATE,EXPRESSION=SSSSSSSSSSSSSSSSSS,BASE=99.

The expression is comprised of constants and keywords. The constant is of the form V'CON' where V = B, O, D, or H implies base 2, 8, 10, or16, respectively, and CON is the constant. The keywords consist of EASY machine registers and scalers, subscripts, operators, and the word BASE. The scalers are indicated as follows:

PC - Program Counter*
ID - Program Identifier Register*
BS - Bottom of Stack*
OBS - Old Bottom of Stack*
BPS - Bottom of Program Space
TPS - Top of Program Space
TS - Top of Stack*

The subscript can be MS, CS, ECB, or I for mainstore, control store, external control block, or indirect, respectively. If a variable is indicated, it will have a value of 0 but can be assigned a value by using the := operator. Allowable variables are X, Y, Z, or BASE. The operators are as follows:

By precedence; highest to the lowest
.LC .LL. .RL. .RA. shift operators
.A. | .V. .X. } logical operators
* / | + - } arithmetic perators
: = (normally only allowed with X, Y, Z, and BASE)

---

* Values returned are those in effect before the server ($COMPU) is executed.

An example is as follows:

```
CALCULATE, (X:=6) + D'64'.
```

The output value is 106 . The default values are no input (blank) for EXPRESSION and 8 for BASE. When no input is indicated, CALCULATE will repeatedly prompt for input.

V.4 CIMPORT

This command is the last step in the package that enables a user to ship card-image files on the CDC 6700 to the QM-1 via the 200UT phone link. (See Appendix G for preceding steps that are not part of the EASY system and see Section V.22 for shipping binary files.)

At the completion of the previous steps, the disk file is in ASCII print format. This command removes extraneous printer controls, etc., that have been added and puts it back into straight card-image format. As soon as the user has brought a file over to the QM-1 using the 200UT command (Section V.56), the user must immediately use the CIMPORT command. If this is not done right away, the file being brought over may be destroyed.

The syntax of this command is:

```
CIMPORT, TO FILE=SSSSSSSSSS,TRIM=AAA.
```

where TO FILE is the name to be given the new file. CEXPORT/CIMPORT supports card images up to 100 characters in length. TRIM allows the user two options, YES (trailing blanks removed) and NO (80-character strings). Defaults are CDR, YES.

V.5 COMPARE

This command compares file A with file B. For disk files, parity is ignored and the number of files must be 1. If two tapes are being compared, then they must both have the same parity. Only six bits per frame are used in the compare of tape files. The leading two bits of each frame from 9-track tapes are ignored. The syntax for this command is:

```
COMPARE,A=SSSSSSSSSS,B=SSSSSSSSSS,NUM FILES=999,PARITY=1,ERRS=SAAA.
```

Errors are written to ERRS (*LPT or *CRT). Defaults are 0,QM1,1,1,*LPT.

V.6 COMPRESS

COMPRESS copies one file to another replacing multiple consecutive blanks with one blank except of course if the blanks occur in a string. Eject, skip and space directives, and comments which occur on one line are deleted. Statements are also concatenated (i.e., lines in the resulting file will contain as much data that will syntactically fit on the line). The syntax for this command is:

```
COMPRESS,FROM=SSSSSSSSSSSS,TO=SSSSSSSSSSSS,NUM OF COLUMNS=99.
```

Defaults are QMR,SCR,72.

V.7 COPY

COPY copies one disk file to another (sequential or word-addressable, but not random access) or card images to a disk file. The syntax for this command is:

COPY,FROM=SSSSSSSSSS,TO=SSSSSSSSSS.

FROM is the source disk file. TO is the destination file. Defaults are *CDR,CDR. (*CDR is the card reader).

V.8 COPYD

The COPYD command is a fast disk copy that will copy one file to another file. The syntax for this command is:

COPYD,FROM=SSSSSSSSSS, TO=SSSSSSSSSS,STOP-ON=AAA.

The user can specify the file to be copied either at end-of-data (EOD) or at end-of-extent (EOE). The defaults are QM1,SCR,EOD.

V.9 COPYNS

This command copies a NOVA file to a SIMPL-Q coded file. The syntax for this command is:

COPYNS,FROM=SSSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=AAAAA.

FROM is the NOVA file. Default is 200UT:LPT. TO is the SIMPL-Q coded file and the default is CDR. STOP ON tells EASY where the NOVA file ends. Default is ETX.

If END is specified, COPYNS stops when a line starting with "END" or ".END" is encountered. (The line containing the END is copied.) The ZEROS option causes COPYNS to stop when a PRU of zeros is read. If ETX is specified, COPYNS stops when an ETX character is read. If EOE is specified, COPYNS will continue until the input file reaches end-of-extent.

V.10 COPYSN

This command copies a SIMPL-Q coded file to a NOVA file. The syntax for this command is:

COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=AAAAA.

FROM is the SIMPL-Q coded file and the default is CDR. TO is the NOVA file and the default is 200UT:CDR.

If the file is to be shipped to the CDC 6000 via 200UT:CDR, specify ETX for END WITH=. If, on the other hand, the file is to be read by the NOVA system, specify ZEROS in order to pad the file with zeros.

V.11 DATE

This command allows the user to set the date (month/day/year). The syntax for this command is:

DATE,=19S79S99. (Default is 01/00/00.)

V.12 DDT

This command invokes the DDT subsystem. DDT is the Dynamic Debug for TRIDENT system. This command is only for use by TRIDENT programmers. The syntax of this command is:

DDT,SCHEMA#=!0!1!2!,UIC=SSS.

This system is described in a separate document, Dynamic Debug for TRIDENT (DDT) User's Guide, NSWC TR 79-96.

V.13 DEADSTART

The DEADSTART command allows the user to modify the system deadstart array. A set of commands is available to the user which includes the following:

·CR·	DIRECTORIES	SET-CSA
·ESC·	EASY-SPACE	SET-BREAKPOINT
·SPACE·	LIBRARIES	

The format of the deadstart array is as follows:

Word	bit: 35 — 18	17 — 0
0	Initial STR	Initial BS
1	TPS	Breakpoint Line
2	Breakpoint	Proc Name
3	3rd Library	Address
4	2nd Library	Address
5	1st Library	Address
6	Mainstore	Limit
7	Control store A limit	Control store B limit
8	1 | CIO 2 | 3	directories 4 | 5 | 6

The EASY-SPACE command allows the user to change the base of stack (BS) and top of program space (TS) to allow varying configurations of EASY. BS should be

no less than 20 while TS must be odd and no greater than the highest main store address accessible to EASY. TS must be at least 44,000 larger than BS to allow sufficient space for EASY.

The STR-BREAKPOINT command is used to set the status and breakpoint registers. An entry of 9999 for BLINE, which is the default value, disables breakpoints. The breakpoint line number should contain the image of the line instruction (i.e., octal line number shifted left logically six bits). The LIBRARIES and DIRECTORIES commands reset the library and directory search orders within the local deadstart array only. The CIO directories contain a six-bit integer indicating drive and user number. (A value of 0-9 indicates drive 0 and 11-19 drive 1.) All 1's are used to indicate a null entry.

The SET-CSA command is used to set the length of control store A segment, placing emulator workspace in it. This is valid only for the configuration of a son.

The carriage return <CR> command performs the deadstart, and the escape <ESC> command allows the user out of the deadstart service without deadstarting. The <space> command displays the latest deadstart table. An example of a deadstart display follows:

```
              DEADSTART PARAMETERS
      BS = 406100        TPS = 747777
 STR = 000000  BID =              BLINE = -1
LIBRARIES:
                1ST           010465 001400
                2ND           000465 001600
                3RD           000205 001400
 CONTROLSTORE A = 037777     MAINSTORE = 01751777
              B = 005777
          DIRECTORIES = 17 19 07 04 03 00
```

V.14 DIRECTORY

The DIRECTORY command allows the user to specify the order in which directories will be searched by the EASY system. The syntax of this command is:

DIRECTORY,SEARCH 1ST=19,2ND=19,3RD=19,4TH DIR=19, REQ DIR=,REQ DIR=.

The user can select a maximum of four directories to search. For each directory specified, the first digit is the drive number (0 or 1) and the second digit is the directory number (0-9). The required system directories 03 and 00 are always searched last. The default values are system directories 03 and 00. Drive 0 assumed if only 1 digit (0 or 1) entered.

V.15 DISK-SAVE AND RESTORE-DISK

The DISK-SAVE command allows the user to selectively save disk files on tape. They may later be restored to disk using the command RESTORE-DISK. The user must be familiar with Appendix D, Disk File Structure on the QM-1, in order to use these commands. If a disk hardware error occurs, an error message number

will be displayed on the CRT. The meaning of the number may be found in the Task Control Program (TCP 2.00) Manual in the Disk Operation section.

The syntax for these commands is:

```
DISK-SAVE, PASSWD=SSSS,MTUNIT=17,MTFILE=99,DSKUNIT=7.
RESTORE-DISK,PASSWD=SSSS,MTUNIT=17,MTFILE=99,DSKUNIT=7.
```

where

PASSWD - a system disk password which the user must know in order to use the command (this password is different from the password needed for locked commands)

MTUNIT - desired magnetic tape drive

MTFILE - desired magnetic tape file number

DSKUNIT - desired disk unit

DISK-SAVE/RESTORE-DISK supports three commands for specifying the disk address of the file to be saved/restored. These are:

```
NOVA-FORMAT
PHYS-FORMAT
USER-FORMAT
```

The NOVA-FORMAT and PHYS-FORMAT commands allow the user to save/restore any specified file residing on the disk. The USER-FORMAT command causes the whole specified NOVA user to be saved/restored.

NOVA-FORMAT

Syntax: NOVA-FORMAT,USER=9,CYL=777,SEC=77,SIZE=777777.

where

USER - desired user (decimal)
CYL - desired cylinder (octal)
SEC - desired sector (octal)
SIZE - file size in NOVA sectors (octal)

PHYS-FORMAT

Syntax: PHYS-FORMAT,CYL=999,HS=99,SEC=99,SIZE=99999.

where

CYL - desired physical cylinder on disk (decimal)
HS - desired head select (decimal)
SEC - desired physical sector (decimal)
SIZE - file size in physical sectors (decimal)

USER-FORMAT

Syntax: USER-FORMAT,USER=9.

where

USER - desired user (decimal)

Upon completion of the appropriate format command, DISK-SAVE/RESTORE-DISK prompts the user to mount a tape on the tape drive and waits for a user response (hit the space bar to continue). It then displays the tape header that is to be associated with the file. This header is of the following format:

```
SV-RES HEADER
DATE =
TIME =
Information on file's disk address
```

The header verifies the file to be operated on to the user. If the user desires to continue, the user hits the return key on the keyboard. Any other response releases the user from the service.

V.16 EDIT

EDIT is a subsystem designed to edit files of SIMPL-Q strings. The syntax for this command is:

```
EDIT.
```

When invoked, the editor will request a file name from the user. If the user inputs a CR only; i.e., a NULL name, the editor starts with an empty file. The file name is any number of characters followed by a carriage return. The contents of the first 23 lines of the file will then be displayed on the screen. A dummy line '***top of file***' is inserted as the first line to inform the user he is at top of the file. After the user finishes editing, the editor will ask for an output file in which to write the final edited file. If a null name, CR only, is given, then the edited version is lost. The input file is modified only if the user gives the name of the input file when an output file is requested.

EDIT operates in one of two modes, Edit or Command, and is initially in Edit Mode.

Edit Mode

Under the Edit Mode, the user can manipulate the data displayed on the CRT with the use of special characters as commands. These commands are explained in the initial display of the EDIT command.

The Edit commands are designed to work on 23 lines of the file displayed on the CRT. These commands are divided into four sets - cursor commands, modify commands, file commands, and miscellaneous commands.

Cursor Commands

Backspace	Move cursor one space to left
Tab	Move cursor one space to right
Control - ^	Move cursor one space up
Line Feed	Move cursor one space down
Carriage Return	Move cursor to start of next line, a carriage return at the bottom of the screen will bring in another line

The cursor commands control cursor movement

Modify Commands

Control - A	Insert line
Control - \	Insert character
Rubout	Delete character
Control - K	Delete line
Control - E	Copy above character
Control - P	Justify paragraph

The modify commands change the text at the cursor location.

The CNTL-A command inserts a blank line by pushing the lines above the cursor up one place. The top line is bumped off the screen.

The CNTL-\ command inserts a space under the cursor and moves the rest of the line to the right. If the line is 80 characters long, an insert will cause the right-most character to be lost.

The RUBOUT command causes the character under the cursor to be deleted.

The CNTL-K command deletes the line under the cursor and pushes up the lines below the cursor to fill the space created.

The CNTL-E command will copy the character above the cursor into the cursor position.

The CNTL-P command starts at the current line and justifies text until a line where the first character is blank is encountered. When CNTL-P is typed, the justify routine begins by moving text into a buffer until the buffer contains more than 80 characters. If the next line to be input is not on the screen, data is first read into an intermediate buffer. It then determines the location where the end of the line is to be placed by beginning at the average margin and scanning in both directions until a blank is found or both minimum and maximum margins have been exceeded. It then outputs the line and deletes it from the buffer. The routine continues to output text until the buffer has less than 80 characters in it. The routine then fills the buffer again. This continues until a line is encountered where the first character is a blank. When this occurs, the routine outputs the remainder of the main buffer and then outputs the intermediate buffer.

File Commands

Control - N	Move down next N lines
Control - F	Find a string
Control - T	Go to top of page
Control - G	Go to line N

The file commands are used to locate the window of 23 lines within the file.

The CNTL-N command will position the cursor to the TOP of the screen and wait for the user to input a number (N). The window is then moved down N number of lines into the file.

The CNTL-F command positions the cursor to the TOP of the screen and allows the user to input a string (up to 80 characters). The file is then searched for the input spring. A message is printed if the file is not found.

The CNTL-T command moves the window to the beginning of the file.

The CNTL-G command positions the cursor to the TOP of the page and waits for the user to input a number (N). The window then will locate line N in the file.

Miscellaneous Commands

Control - R	Rewrite screen
ESC (key)	Enter command mode
ESC?	Display commands
ESC<COMMAND>?	Explain commands

The CNTL-R command restores the screen. This is used if the screen contents are lost due to static or operator error (using the local edit keys, or erase page will mess up the screen).

The ESC (key) command causes the editor to enter Command Mode. A prompt (!) will appear at the bottom of the screen and the user may enter commands via master.

The user can see a display of the available commands in the Command Mode by pushing the ESC (key) command followed by a ?. The ESC (key) command followed by <COMMAND>? will explain the specified command.

Command Mode

The Command Mode uses MASTER and supports the following commands:

CHANGE - change occurrences of one string to another string
GETFILE - insert specified file contents into edit file
HELP - display available commands
QUIT - leave editor and save edit file

```
RECOVERY    - recover temporary editor file
SET-MARGIN  - set margin parameters
TOP         - go to top of file
WRITE       - write out lines from edit buffer
<@>         - return to edit mode
<ESC>       - return to edit mode
```

CHANGE

SYNTAX: CHANGE,OPTIONS=AA

DEFAULT: V (Veto on 1 occurrence)

This command changes occurrences of one string to another string. The command prompts for the FROM/TO string. Options are V (veto), G (global), VG, GV. Search starts at the current cursor position. Global will change all occurrences from the current cursor position to the end of file. Veto allows the user control over each change by printing each change and asking the cursor for a Y, N, Q, or G. Y (yes) causes the change to be made and N (no) causes the change not to be made. When veto is used with global (VG or GV), Q will cause the Change command to halt and G will cause the veto option to stop and finish all changes. Change will print all changed lines.

GETFILE

SYNTAX: GETFILE,FILE=SSSSSSSSSS,FROM LINE=999999,TO LINE=999999.

DEFAULT: ,1,999999.

GETFILE will ask for a file name and the number of lines. This command will insert the number of lines from the named file into the edited text following the line the cursor is on. If the number of lines in the file is less than the number requested it will insert all lines in the file and return to the user.

HELP

SYNTAX: HELP.

This command will cause a display of the screen-oriented Edit Mode commands.

QUIT

SYNTAX: QUIT,OUTPUT FILE=SSSSSSSSSS,TRIM=A.

DEFAULT: ,Y. (Default is to not save changed file)

QUIT will ask the user for a file on which the edited text is to be written. If no file name is given, then the edited text will be lost. The original input file (if any) will not be changed unless it is named as the output file. A Y (yes) will trim the output and an N (no) will output 80-character strings. This command will return the user to the EASY level.

RECOVERY

Syntax: RECOVERY.

EDIT uses two temporary files (STEXT and SQUAD). EDIT normally reads the input file, and displays one screen full of data. As the old lines are removed from the working screen, they are copied to one of the two temporary files (in ping pong fashion). Any backward motion of the current screen (e.g., control "T"; TOP) will cause the rest of the input file to be written out to the current temporary file. Then, the temporary file is rewound and used as input. This time, the other temporary file is used as the new output file.

If a problem (e.g., crash, mistake, etc.) occurs while editing, the user can recover from the temporary files as follows:

1. Type EDIT (EDIT will ask for an input file).

2. Type "ESC" key.

3. Type "RECOVERY" - If this is the final recovery attempt, then the system copies the two temporary files to two other temporary files (STEXT→STOK, SQUAD——→SDATA). The system displays a message about copying these files. EDIT will use these files for input one at a time in ping-pong fashion. The first file selected is STOK, the old STEXT file. EDIT informs the user as to which file is being used and returns to Edit Mode. NOTE: The time and date when the file was created is displayed as the first line (e.g., '***TOP OF FILE***<current output date/time><recovered version date/time>'). The user can now use this file for normal editing. If for some reason the user would like to examine the other temporary file, the user can do so by typing the "<ESC>" key and RECOVERY again. The user examines the second file and/or ping pong by " ESC " RECOVERY back to the first temporary file. The user should be careful not to leave EDIT (e.g., EXIT,CNTRL-Z etc.) until the desired temporary file has been copied (using the <ESC> Q command) to a user file.

If the user exits EDIT with the <ESC> Q command after doing editing and fails to save his file, then an automatic top (<CNTRL>-T) is performed. Therefore, the user could use the above recovery option to obtain his edited file.

SET-MARGIN

Syntax: SET-MARGIN,AVERAGE=99,MINIMUM=99,MAXIMUM=99.

Default: 60,1,80.

This command allows the user to set margin parameters for the justify (CNTL-P) command in the Edit Mode.

TOP

Syntax: TOP.

This command will go to the top of the file and return the user to the screen oriented Edit Mode.

WRITE

Syntax: WRITE,FILE=SSSSSSSSSS,LINES=9999,KILL=A.

Default: QM1,0,N.

This command will write out lines from the edit buffer to the output file. If KILL is specified, the lines are removed from the edit buffers. If there are no lines left in the edit buffer, the output file is padded with blanks.

<@>

Syntax: <@> . Terminate Command Mode and return to screen-oriented Edit Mode

<ESC>

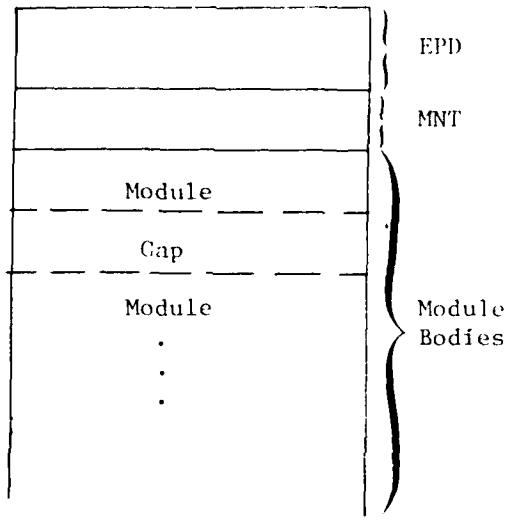
Syntax: <ESC>. Terminate Command Mode and return screen-oriented Edit Mode.

V.17 EDITLIB

EDITLIB is a subsystem of EASY which manipulates EASY program (binary) libraries. These libraries are those referenced by the PSLOAD function of EASY. (See SIMPL-Q Reference Manual and EASY - The Design and Implementation of an Intermediate Language Machine for details of PSLOAD.)

Library Structure

The library consists of three major parts: the Entry Point Directory (EPD), the Module Name Table (MNT), and the module bodies. The EPD is a hash-coded table of entry points, used by the PSLOAD function. The MNT is a table of modules and gaps. Following these tables are the modules themselves, each starting on a PRU boundary. Current configuration is 256 MNT slots (maximum of 256 modules) and 2,048 EPD slots (64 has bins x 32 slots per hash bin).



Running EDITLIB

The syntax of this command is:

EDITLIB,LIBRARY=SSSSSSSSSS,STATUS=AAA.

where

LIBRARY - name of the library
STATUS - OLD or NEW

If OLD status is specified, the file should contain good EPD,MNT, and module bodies. If NEW is specified, the file may be garbaged, and the only permissible command is then BUILD. To exit from EDITLIB, the user uses the single key command @. While in the EDITLIB subsystem, the EASY commands BIND, EXEC, and MT-TO-DISK are available to the user.

EDITLIB supports the following commands to manipulate these libraries:

1. If status of library was specified as OLD -

   ADD - add modules to library
   REPLACE - replace module in library
   REMOVE - remove module from library
   DELETE - delete entry from the directory (EPD)
   LIST - listing of library
   SMASH - packs the module-bodies
   COPY-MODULE - copies modules from the library to a given file

2. If status of library was specified as NEW -

   BUILD - build a new library
   INCLUDE - specifies module to go into new library being built

ADD - adds one or more modules to the library.

```
ADD,FILE=SSSSSSSSSS,# MODULES=999.
```

Defaults are QM-1 for file name and 1 for the number of modules. If a module by the same name already exists on the library, it is left intact and a message is displayed. If any entry points in the module duplicate any others in the library, the module is not added. If more than one module is to be added, they should be WELDed together on the CDC 6000 (see SIMPL-Q Manual). NOTE: Two entry points by the same name may coexist on the same library if they are different types (i.e., one procedure, one data item).

REPLACE - replaces one or more modules in the library.

```
REPLACE,FILE=SSSSSSSSSS,# MODULES=999.
```

Defaults are QM-1 and 1. If a module by the same name does not already exist on the library, the new one is added and a message is displayed. Entries in the old module being replaced are deleted before the new module's entries are added.

REMOVE - removes a module from the library.

```
REMOVE,MODULE=ANNNNN.
```

All of the module's entries in the library are deleted. Do not use the default module name.

DELETE - deletes an entry from the directory (EPD).

```
DELETE,ENTRY=ANNNNN,TYPE=A.

ENTRY - entry point name
TYPE  - D for data
        P for procedure
```

The entry point is deleted from the EPD but this does not delete the entry marker from the module itself. Default (X88888,P) is dummy entry generated by BIND.

LIST - prints a listing of the library.

```
LIST,LEVEL=7.
```

LEVEL is a number from 1 to 5 indicating the level of information to be printed. Default is level 2. As the level number increases, more information is added to the printout.

Level	Information
1	alphabetized list of module names and lengths
2 Level 1 plus	alphabetized list of entry points, with types and names of modules
3 Level 2 plus	alphabetized list of modules with associated entry points and types
4 Level 3 plus	table loading statistics (EPD and MNT)
5 Level 4 plus	module body map, including gaps and gap statistics

SMASH - packs the module bodies.

SMASH.

This packs (compresses) the module bodies eliminating all gaps (like a trash compactor). This should only be performed when the library is running out of space and there are a lot of small gaps.

COPY-MODULE - copies modules from the library to a given file.

COPY-MODULE,TO FILE=SSSSSSSSSS.

EDITLIB repeatedly prompts the user for the name of a module, and more than one module may be copied onto the file. Default is file QM-1. The library is not changed by this command. To leave this command, enter a <CR> for a module name.

BUILD - builds a new library.

BUILD,PASSWORD=SSSSSSSSSS.

BUILD builds a library initially, after the user supplies a special password. BUILD clears space on the disk for the EPD and MNT, then calls on MASTER to wait for a number of INCLUDE commands (description follows) from the user (i.e., BUILD is actually a subsystem of EDITLIB). After the INCLUDEs, BUILD builds the EPD and writes both tables to disk. The library status automatically becomes OLD after successful completion of a BUILD. It is possible to use BUILD if library was specified as OLD, but generally should be used after a specification of NEW.

INCLUDE - specifies module to go into new library being built.

INCLUDE,FILE=SSSSSSSSSS,# MODULES=999.

Defaults are QM-1 and 1. The module names are checked against those already in the MNT (from preceding INCLUDEs) and if not found, are entered into the MNT and the module body is placed in the library. The entry points are ignored at this time. If more than one module is in the file, they must have been WELDed together on the CDC 6000 (see SIMPL-Q Manual).

When the user has done all the INCLUDEs, a signal of completion is used by the single key command "@". This causes to complete BUILD its job and build the EPD and then exit from the BUILD subsystem of EDITLIB.

V.18 EDITOR

Editor is a subsystem designed to edit SIMPL-Q source programs on a line-by-line basis. Commands are provided that allow the user to add, delete and/or change selected lines or portions of lines, to search for specific strings of text, and to print selected lines of text. It is an in-memory editor and can presently handle 20,000 characters in 800 lines of text. A line of text can be a maximum of 80 characters in length.

The syntax for this command is:

```
EDITOR.
```

The Editor operates in one of two modes, File or Edit, and is initially in File Mode. Edit Mode has two submodes, Input and Command. The File Mode commands OLD, NEW, and EXISTENT will initiate Edit Mode. While in Edit Mode, an "@" will initiate File Mode. While in File Mode, an "@" will terminate the Editor service and the contents of the edit buffer are lost unless they have previously been saved.

File Mode

File Mode supports two types of files, PDS (partitioned data set) and non-PDS. A non-PDS file is a file containing card image strings. A PDS file is a file that is partitioned into subfiles called partitioned data sets. The PDS file has a file name of six characters and each partitioned data set in it also has a six-character name. Each PDS is actually a SIMPL-Q string in a special format for the editor (i.e., similar to card-image strings of the non-PDS file). Each PDS may contain a module or program to be edited. File Mode uses MASTER and supports the following commands:

```
OLD         - reads in disk file and invokes Edit Mode
NEW         - empties edit buffer and invokes Edit Mode
EXISTENT    - invokes Edit Mode with existing edit buffer
SAVE        - saves current edit buffer on disk
DELETE-PDS  - deletes partitioned data sets from a PDS file
CREATE-PDS  - restructures a non-PDS file into a PDS file
GARBAGE     - does a garbage collection on a PDS file
```

OLD - reads in disk file and invokes Edit mode.

```
OLD,FILE=SSSSSS,PDS=SSSSSS.
```

where

```
FILE - name of file, PDS or non-PDS. Default is SOURCE, a PDS file.
PDS  - name of partitioned data set if PDS file. Default for PDS is
       six blanks. Use the default if non-PDS file.
```

This command reads the contents of the specified file or partitioned data set into the edit buffer and invokes Edit Mode.

NEW - empties edit buffer and invokes Edit Mode.

NEW.

This command is used when the user wishes to create new text. It empties the edit buffer and invokes Edit Mode.

EXISTENT - invokes Edit Mode with existing edit buffer.

EXISTENT.

This command allows the user to reenter Edit Mode with the current contents of the edit buffer. For example, the user can leave Edit Mode and go to File Mode to save a copy of the edit buffer and then return to Edit Mode with this command to continue editing.

SAVE - save current edit buffer on disk.

SAVE,FILE=SSSSSS,PDS=SSSSSS.

where

FILE - name of file, PDS or non-PDS. Default is SOURCE, a PDS file.

PDS - name of partitioned data set if PDS file. Default for PDS is six blanks. Use the default for non-PDS FILE.

This command stores the current edit buffer in the specified PDS or non-PDS file. This file must already exist on the disk.

DELETE-PDS - deletes partitioned data sets from a PDS file.

DELETE-PDS,FILE=SSSSSS.

where

FILE - name of a PDS file

This command will prompt the user for the six-character name of the PDS to be deleted. This prompting for a PDS name to delete will cycle until the user responds with "@" to terminate the deletion process.

CREATE-PDS - restructures a non-PDS file into a PDS file.

CREATE-PDS,FILE=SSSSSS.

where

FILE - name of the non-PDS file

This command which restructures a non-PDS file into a PDS file is locked. There is no facility in the EASY system for dynamically allocating disk files. Before a file can be used in the EASY system, it must be preallocated using the stand-alone NOVA emulator. The file named in this command must already exist on the disk. This command will then turn it into a PDS file. Data may be placed in the file using the SAVE command.

GARBAGE - does a garbage collection on a PDS file.

GARBAGE,FILE=SSSSSS.

where

FILE - name of a PDS file

The user should use this command when a file error occurs during a SAVE on a PDS file. Invocation of this command on a non-PDS file can result in loss of data in the non-PDS file'! The garbage collection does not damage the edit buffer contents.

Edit Mode

The Edit Mode does not use MASTER for processing commands. Whenever it is ready for a new command, it will prompt the user with an asterisk (*). Edit Mode operates in one of two submodes, Input or Command. It is initially in Command Mode.

The Edit commands are described in terms of a viewing window. The size of this window is one line. Each command moves the viewing window through the text in some manner. If an attempt is made to move the window off the bottom of the text, an END OF BUFFER message results. When an END OF BUFFER condition arises, the window is positioned below the last line of text and is empty; this allows insertion of new text at the end.

It is also possible to attempt to move the window up off the top of the text. In the case, the window will be positioned above the first line of text and will again be empty, allowing text to be inserted ahead of old text. This is a LINE ZERO condition. The TOP command always moves the window to LINE ZERO.

Input Mode

Input Mode is to be used for entering several consecutive lines of text. The text is added just below the current position of the window. Therefore, the window should be positioned to the desired location (using one of the commands in Command Mode) before entering Input Mode if the file status was declared OLD. If the file status was declared NEW, no text exists in memory and the window does not need to be positioned.

To enter Input Mode or to leave Input Mode (i.e., return to Command Mode), the user enters a null line (carriage return only). While in Input Mode, the user types in as many lines of text as he desires (one after the other), ending each with a carriage return. There is no prompting in Input Mode.

Command Mode

Command Mode supports the following commands:

Top	- move window to top of text
Print	- display lines on CRT
Next	- move window down specified number of lines
Up	- move window up specified number of lines
Delete	- delete lines
Locate	- locate specified string of characters
Change	- change selected occurrences of a given character string to a different string
Insert	- insert single line of text below window
Add	- insert contents of non-PDS file below window
Kopy	- copy from window to end of buffer to non-PDS file

In general, an Editor command consists of the command specification followed by optional blanks, followed by parameters that further qualify the action of the command and terminated by a carriage return. The command specification is the first letter of the command name. Commands must begin in "column 1" (i.e., immediately after the * prompt). A summary of the commands available in Command Mode may be obtained by typing "?" (followed by carriage return) after the * prompt. In Input Mode, "?" is just an input character.

Top

Syntax: T

The Top command has no parameters; it moves the window to the top of the text, ahead of the first line of text. After a Top command, a LINE ZERO condition results.

Print

Syntax: P n

where

n - number of lines to be printed

The Print command prints the specified number of lines on the CRT, moving the window downward as it does so. The first line printed will be the current line; no printing will result if there is no current line (i.e., the window is empty), but the empty window will be counted in the number of lines specified. After a print, the window will be positioned over the last line printed. Thus, a 2P command does not move the window. If the number of lines is not specified, one (1) is assumed.

Next

Syntax: N m

where

m - the number of lines

The Next command moves the window down throughout the text; it takes as a parameter the number of lines to be moved. For example,

N 10

moves the window down 10 lines.

If the number of lines to be moved downward exceeds the number of lines remaining, the END OF BUFFER condition results; i.e., the window is positioned below the last line of text and is empty. When used in place of a number (as above), the asterisk character is equivalent to a very large number. For example,

N *

moves the window to the end of the buffer.

If the number of lines is not specified in the command, one line is assumed. Once the Next command has moved the window, it will then display the new window contents.

Up

Syntax: U n

where

n - the number of lines

The Up command moves the window up the specified number of lines. If no number is specified, one (1) is assumed.

Delete

Syntax: D n

where

n - number of lines

The Delete command deletes the specified number of lines, starting with the current line and going downward. Thus,

D 5

will delete the current line and the four which follow it; after a Delete the window will be at the line following the last Delete. Note that the current window contents are always counted as the first line deleted.

If the number of lines to be deleted has not been specified, one (1) is assumed.

Locate

Syntax: L string

The Locate command searches through the text downward for an occurrence of the specified string of characters. The search starts at the line after the current line and continues until the string is found or an END OF BUFFER condition arises. When the string is found, the line containing it is printed, and the window is positioned over that line.

If a Locate is given while at the END OF BUFFER, the Editor will assume an implied Top command and start searching downward from the beginning of the text.

If string is not specified, the Editor will use the string it used on the last Locate command.

Change

Syntax: C/old-string/new-string/

The Change command is one of the most important commands, allowing a change from selected occurrences of a given character string to a different string. Basic options on the Change command allow selective changing of:

1. All occurrences on a line of a particular string, or only the first such occurrence.

2. Several lines at one time, or only the current line.

In using the Change command, one first specifies the string of characters to be changed, and then the string of characters it should be changed to. This is done by typing the two strings (old first) enclosed by an arbitrary delimiting character; the delimiter used will be the first nonblank character after the command name. Blanks are significant in both the 'old' and 'new' strings. As an example, consider changing the word SINF to SINE in the current line. One would type:

C;SINF;SINE;

In this case, the semicolon is used as the delimiter.

Either the old or the new string may be null; i.e., contain no characters. Thus, one could entirely remove the word GLITCH from a line by entering:

C#GLITCH##

Notice that in the above example the delimiter is a number sign. The word HERE could be inserted at the beginning of a line by typing

```
C//HERE/
```

The above examples of the Change command will change only the first occurrence of the given string on the line. One can indicate that all occurrences on a line are to be changed by typing a G (for GLOBAL) after the edit string. Thus,

```
C/XX/XY/ G
```

would change all occurrences of the string XX to XY on the current line.

The Change command in either of the above forms may be directed to operate on several lines by typing the number of lines to be changed. Thus,

```
C;ABC;AB; 3
```

will change the first occurrence (if any) of ABC to AB in each of three successive lines, starting with the current line, and

```
C/3.1416/3.1415926/ 100 G
```

would change all occurrences of 3.1416 for 100 lines; i.e., the current line and the 99 which follow it.

The new (i.e., changed) version of the line will be printed for each changed line. One may take advantage of this feature to find all occurrences of a given string by changing it to itself. The commands:

```
T
C:2*3:2*3: 10000
```

would cause all lines containing the string 2*3 to be printed.

The position of the window after a Change command will be at the last line which was examined for changes, regardless of whether it was changed or not. Since the first line examined is the current line, this means that

```
C/A/B/ 5
```

will leave the window positioned as if one had typed:

```
N 4
```

If only C is typed, the Editor will use the "old string" and "new string" from the last Change command.

Insert

Syntax: I text

The Insert command allows the insertion of a single line of text immediately below the window, without having to toggle to Input Mode then back to Edit Mode again. The command:

I THIS IS TEXT

will insert the line:

THIS IS TEXT

Add

Syntax: A filename

This command will insert the contents of the specified non-PDS file immediately below the window.

Kopy

Syntax: K filename

This command will copy from the window to the end of the buffer to the specified non-PDS file.

V.19 EXEC

This command enables the user to switch MASTER from one input medium to another. The default input medium for MASTER is the keyboard. The EXEC command allows the user to have commands executed from a file (cards or disk) instead of the keyboard, thus implementing a "batch" capability.

The syntax for this command is:

EXEC,FILE=SSSSSSSSSS.

The default for FILE is *CDR - input medium is to be cards.

EXEC files (from which MASTER is to execute commands) are card image files, with one command per card. Keywords must be fully spelled out, commas placed between parameters, and a period at the end. One-key immediate-execute commands (e.g., @) are not recognized on EXEC files,so their full names must also be spelled out (e.g., <@>).

EXECs are not stacked, and there is no inherent hierarchy in them. If one EXEC file has an EXEC command in it, MASTER switches to the new file and rewinds the old one.

MASTER switches back to the keyboard:

1. At the end of an EXEC file,

2. Upon a syntax error in an EXEC file,

3. If a command in an EXEC file aborts.

The following sample EXEC file builds a new library.

```
MT-TO-DISK,,,JUNK.
EDITLIB,SYSTEM,NEW.
BUILD,PASSWORD.
INCLUDE,JUNK,41.
<@>.
<@>.
```

V.20 FH-UTILITIES - FILE HANDLER UTILITIES

FH-UTILITIES.

This command invokes the Pseudo Dynamic File Management System which was designed to handle file requirements of the Dynamic Debug for TRIDENT (DDT) project. This allows for the dynamic creation, deletion, and manipulation of files without the requirement to enter the NOVA simulation mode. The FH-UTILITIES makes use of a large preallocated file on disk. Dynamic Pseudo Files within the large preallocated file are then manipulated by the various file handler commands.

The file handler commands are part of the CMDFH command file. The templates are listed in Appendix C. The commands are as follows:

<@>	FIND-ENTRY	LIST-DIR	RET-STR
CREATE	G-COLLECT	LST-ENT-INFO	UPDATE
DEL-ENTRY	INITIALIZE	OPEN	

<@> - terminates the file handler.

CREATE - creates a file by calling procedure FHCRE8E.

```
CREATE,ENTRY-NAME=SSSSSSSSSS.
```

DEL-ENTRY - deletes a file entry by calling procedure FHDELE.

```
DEL-ENTRY,ENT-NAME=SSSSSSSSSS.
```

FIND-ENTRY - verifies that a specific file exists procedure FHFINDE is called to perform the search.

```
FIND-ENTRY,ENT-NAME=SSSSSSSSSS.
```

G-COLLECT - performs the garbage collection function of file handling. It deletes file information no longer needed and compacts the remaining information.

G-COLLECT,FILE-NAME=SSSSSSSSS.

INITIALIZE - calls FHINIT to initialize the file handler facility. It should be used prior to using the other file handler initialization routines.

INITIALIZE,FILE-NAME=SSSSSSSSS.

LIST-DIR - lists the file directory.

LIST-DIR.

LST-ENT-INFO - lists specific entries of a file.

LST-ENT-INFO,INDEX=9999.

OPEN - opens a file.

OPEN.

RET-STR - returns a string of records starting at the specified index number.

RET-STR,INDEX=9999,REC #=9999.

UPDATE - updates a file. The operator can be D for delete or R for replace.

UPDATE,INDEX=9999,OPR=A,OLD=SSSSSSSSSSSSSSSSSS,
NEW=SSSSSSSSSSSSSSSSSS.

V.21 FILES

FILES is a subsystem for manipulating directories. The working directory is established when the FILES processor is invoked. The syntax for this command is:

FILES,FIRST USER=19. Default is 04.

A user directory is specified by a binary digit denoting the disk unit, followed by a decimal digit denoting the desired user on that pack.

FILES uses master and supports the following commands:

ADD	- adds new file to directory
CHANGE	- changes file attributes
COMPACT	- compacts current user (LOCKED)
DELETE	- deletes specified file

```
DISPLAY    - displays directory entry for a file
INITIALIZE - clears directory (LOCKED)
LOCK       - locks console (LOCKED)
MOVE       - moves file to new location (LOCKED)
OVERLAPS   - displays overlaps
REPORT     - displays current directory
UNLOCK     - unlocks console
USER       - change working directory
<@>        - terminate FILES
```

ADD

Syntax: ADD,FILE NAME = SSSSSSSSS,LENGTH=77777.

Default: $,300. ($ illegal file name)

This command adds a new file to the working directory. The length (extent) of the new file is expressed in terms of NOVA sectors, and files of zero extent are permitted (these will be placed immediately after the directory). FILES will perform a first-fit search of the gaps in the user to find a place to put the new file so that it will not overlap any previously existing file. Possible errors from this command are "illegal file name," "duplicate file name" (in the current user), "insufficient space for file," and "no available directory slots." The end of data for the new file is automatically set to zero and the permission for the file is set to read/write.

CHANGE

Syntax: CHANGE,FILE=SSSSSSSSS,NEW NAME=SSSSSSSSS,
EODPRU=77777,EODWORD=177,PERM=A.

Default: $,*,*,0,*. ($-illegal file name)

This command allows the user to change a file's name, its end of data, and/or its permission. If the new name, the EODPRU, or the permission is specified by a "*" (default), then the corresponding attribute is left unchanged. The possible error messages are: "illegal file name" (old or new), "file does not exist" (old name), "duplicate name" (new name), and "EOD exceeds file's extent." If one of these occurs, the command is aborted and the directory is not changed in any way.

COMPACT

Syntax: COMPACT,PASSWORD=SSSSSSSSS.

Default: WRONG. (Illegal password)

This command compacts the files in the current NOVA user so that all of the gaps will be squeezed to the end. This involves rewriting the directory and possibly extensive data transfer. If a disk hardware error would occur during the execution of this command, the state of the NOVA user would be questionable. Therefore, the user should perform a DISK-SAVE first. In order to access this command, the correct password must be typed by the operator. Then a warning

message (about the hazards of hardware error) is displayed. The command proceeds if the operator types "GO". At this point the new directory will be written and the files will be moved downward as necessary. Overlapped files are moved together to preserve their relative positions. This command is locked.

DELETE

Syntax: DELETE,FILE NAME=SSSSSSSSSS.

Default: $. (Illegal file name)

This command finds the directory entry for the named file and displays it on the CRT. The user is then asked if he really wants to delete the file. If he types a "Y", the file is deleted, but if any other key is struck, the command is aborted. An error message results if the file name is illegal or if the file does not exist.

DISPLAY

Syntax: DISPLAY,FILE NAME=SSSSSSSSSS.

Default: $. (Illegal file name)

This command displays the directory entry for a single file upon the CRT. If the file does not exist or if the file name is illegal, an error message to that effect is displayed.

INITIALIZE

Syntax: INITIALIZE,USER=19,PASSWORD=SSSSSSSSSS.

Default: XX,WRONG. (Illegal)

This command clears the specified directory; any files in it will be lost. To access this command, the proper password must be entered. A warning message about the effects of this command will be displayed, to which the operator must respond with a "GO". The current directory will be displayed on the CRT. The operator will then be asked if he wants to destroy all files in the directory. If he responds with "YES", the directory will be cleared to zeros (empty). A single file, named "DIRVn" (where n is the decimal digit for the user) will be created. It will start at (NOVA) cylinder 0, sector 0, for an extent of 110 octal NOVA sectors. Its EODPRU will be 60 octal, EODWORD will be 0, and the permission will be read/only. This file represents the reserved space at the beginning of the user. The directory (containing only one file) will then be written to disk, completing the initialization. This command is locked.

LOCK

Syntax: LOCK.

Locks console for protection of locked commands.

MOVE

Syntax: MOVE,FILE=SSSSSSSSSS,START CYL=777,SECTOR=77,
LENGTH=77777,MOVE DATA =AAA.

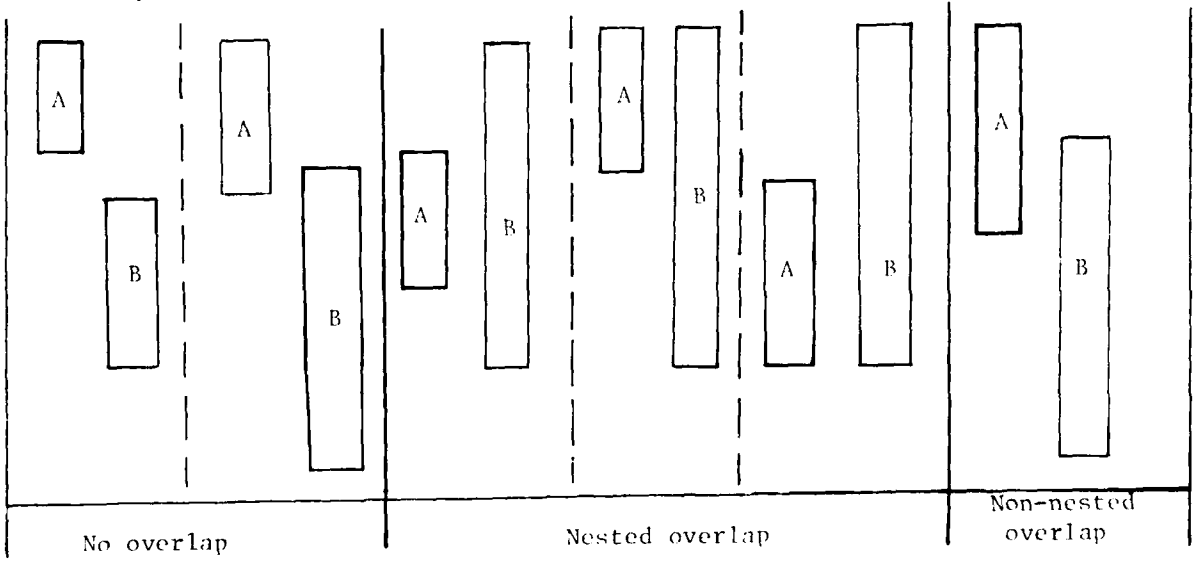
Default: $,*,0,*,NO. ($ - illegal file name)

This command allows the user to place the file anywhere within the current NOVA User disk space by specifying the starting address (NOVA cylinder and sector) and extent (NOVA sectors). If the starting cylinder or the length is specified as a "*" (default), then the corresponding attribute (start address, extent) is left unchanged. The overlaps which will occur if the move is accomplished will be displayed to the user, and then he is asked if he really wants to perform the move. If the "Y" key is struck, the move will be performed, but if any other key is struck, the command will abort with no changes made. If the file extent is decreased below the end of data, the end of data will be set to the end of extent and an information message will be issued. If the file name is illegal or the file cannot be found, an error message will be displayed and the command will be prematurely aborted. In case of a successful move, the actual data in the file will also be moved if the user specifies "YES" in the command line (not the default) and the starting address has been changed. The amount of data to be moved will be the minimum of the old extent and the new extent. This command is a locked command.

OVERLAPS

Syntax: OVERLAPS.

This command displays all cases of overlapping files in the current user. There are two types of overlaps (see figure below): nested and non-nested. Nested overlap occurs when one file is completely contained within another file. Non-nested overlap occurs when two files partially overlap, but each file contains space not contained in the other file.



REPORT

Syntax: REPORT,TO=AAAAAAA,SORTED BY=AAAAAAAA,SUBFILES=AAA.

Default: CRT,POSITION,YES.

This command will display the current directory. All of the active entries in the working directory are displayed upon the CRT, PRINTER, or BOTH. The current date and time, user number, and number of empty (inactive) directory entries is also displayed. The file entries may be sorted either by NAME or by POSITION on the disk; if the positional sort is selected, the gaps (unused space, available for allocation) are also displayed. The user may select whether or not subfiles (files nested within another file) are to be shown in the display.

UNLOCK

Syntax: UNLOCK,PASSWORD=AAAAA.

Default: WRONG. (Illegal)

If the proper password is entered, access to the locked commands (MOVE,COMPACT,INITIALIZE)is permitted.

USER

Syntax: USER,CHANGE TO=19.

Default: *. (Displays current user)

This command will change the working directory. If the default is used, the current working directory is displayed.

.@.

Syntax: <@>.

This command terminates the operation to the FILES subsystem.

V.22 IMPORT

This command is the last step in the package that enables the SIMPL-Q programmer to compile programs on the CDC 6700 and ship the binaries directly to the QM-1 disk via the 200UT phone link, eliminating the use of tapes. See Appendix F, Export/Import User's Guide for preceding steps which are not part of the EASY system. (See Section V.4 for shipping card image files.)

At the completion of the previous steps the disk file is in an ASCII print format because the 200UT does not support binary files. This command transforms the file back to its binary form. As soon as the user has brought a file over to the QM-1 using the 200UT command (Section V.56), the user must immediately use the IMPORT command. If this is not done immediately, the file brought over may be destroyed.

The syntax of this command is:

IMPORT,TO FILE=SSSSSSSSSS.

where

TO FILE - name to be given the new file. Default is QM1.

V.23 LIBRARY

The LIBRARY command allows the user to select up to three system libraries from which EASY loads unsatisfied externals. The libraries are searched in the order specified. The syntax of this command is:

LIBRARY,SEARCH 1ST=SSSSSSSSSS,2ND=SSSSSSSSSS,3RD=SSSSSSSSSS.

SYSTEM is the default value for each library.

V.24 LISTCF

This command lists a command file on the printer. The header is printed first, followed by the list of keywords in the command file. The templates are then printed in alphabetical order.

The syntax for this command is:

LISTCF,COMMAND FILE=SSSSSSSSSS.

The default file name is CMDTMP (the command file for the EASY system).

V.25 LOCK

This command returns the EASY SYSTEM to "locked" (i.e., unprivileged) command mode. The system header is rewritten to reflect the mode change ('UNLOCKED' replaced by 'EASY SYSTEM') and the system flag "UNLOCKED" is set to false.

The syntax for this command is:

LOCK.

This command is "locked" (i.e., may only be used in privileged mode of operation).

V.26 MACRO

This command invokes the SIMPL-Q standalone macro processor. The macro processor is described in the SIMPL-Q Manual.

The syntax for this command is:

MACRO,OPTIONS=AAAAAA,INPUT=SSSSSSSSSS,OUTPUT=SSSSSSSSSS.

OPTIONS - N - Do not list input element
S - List input element
Z - List processing time

The options may be combined in any meaningful way. The default is S. For no listing use N.

INPUT - Name of input file. The default is *CDR, the card reader. If input is a disk file, it must be in card-image format.

OUTPUT - Name of output disk file - default is CDR.

V.27 MAKECF

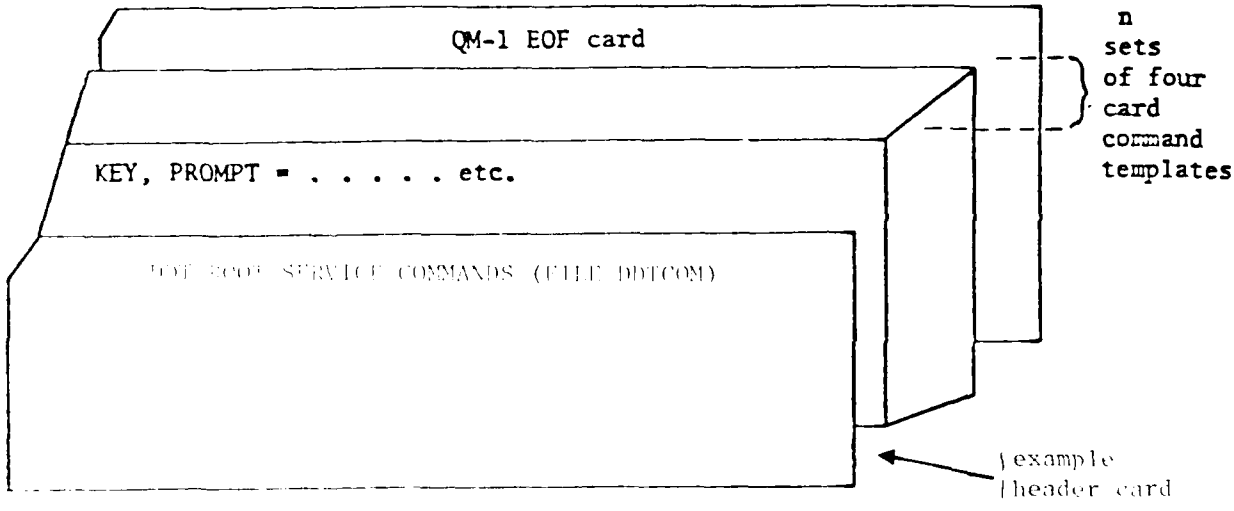
This utility accepts a card deck (or disk file) in the command deck format and generates a command file of standard format.

The syntax for this command is:

MAKECF,INPUT=SSSSSSSSSS,COMMAND FILE=SSSSSSSSSS.

The default file names are *CDR and CMDTMP (*CDR is the card reader).

Example Command Deck:



Any given command deck cannot hold more than 32 templates, and naturally, no duplicate keywords. The "header card" prefixes the list of acceptable keywords displayed in response to an input of "?" by the EASY user. Template format and command file format are defined in Sections II.4 and II.5.

V.28 MAKERESIDENT

This command will allow the user to bring in up to eight modules referenced by entry names. MAKERESIDENT will display the main store location of the entry names. The syntax for this command is:

MAKERESIDENT,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,
=SSSSSS.

The defaults are =DBUGR, DBGSUP, MAP, STEPEZ, STRACE, $FAULD, which makes CONTROL-Q resident. Any modules made resident will be returned when an <@> is typed.

The following example illustrates the use of MAKERESIDENT:

```
!!MAKERESIDENT,=DBUGR,=MAP,=,=,=,=,=,=.
  ENTRY  <DBUGR>  RESIDENT AT 657746.
  ENTRY  <MAP>    RESIDENT AT 654360.
```

V.29 MODULATE

MODULATE,BINARY=<input file name>,FORMAT=<input file format>,
TO=<output file name>,NAME=<module name>.

This function encapsulates an emulator, which normally resides on two files (A-segment and B-segment), for ingestion by EDITLIB. This command is normally used in pairs, one MODULATE for the A-segment and one for the B-segment. The binary file name is specified by file name. If none is specified, default prompts for additional file names.* The format is the format of the microcode generated by the assembler/compiler. NCS format is the format for binary microcode as generated by the Nanodata microassembler.

The input files are converted into SIMPL-Q modules. A module has a header indicating creation by computer version '0.0' (indicating not created by a compiler) and a compile date and time equal to the date and time known by EASY when the file was modulated. The module name is a 1 to 6 character name starting with a letter. The file name it is written on is a 1 to 10 character disk file name. The convention for naming these files consists of mnemonics identifying the target machine followed by 'WS' or 'EM' (where 'WS' indicates working storage or control-store A-segment and 'EM' indicates emulator which resides in the control-store B-segment). The modules may then be edited, using EDITLIB, into any library under EASY and loaded by EASY automatically when referenced.

V.30 MT-TO-DISK

MT-TO-DISK copies the binary output produced by the SIMPL-Q compiler on the CDC 6700 from tape to disk on the QM-1.

The syntax for this command is:

MT-TO-DISK,UNIT=17,FILE #=99,TO FILE=SSSSSSSSSS,REWIND=SSSSSS.

---

* The META and SMITE capabilities are not yet implemented.

The UNIT parameter specifies the tape unit number (default is 0). FILE # specifies the number of files to space forward before copying begins (default is 0). TO FILE specifies the disk file name (default is QM-1). REWIND allows the user four options, BEFORE (rewind before file searching begins), AFTER (rewind after copy), BOTH (rewind before and after), and NONE (no rewind). Default is NONE. Tape records must be 1,536 frames long with six good bits per frame. (This is format of 6700 SIMPL-Q compiler output.) Tape records will be reformatted into 256 SIMPL-Q word (i.e., 36 bits) blocks and written sequentially to disk.

V.31 PASCAL - UCSD PASCAL EMULATION SYSTEM

PASCAL,FLOPPY=SSSSSSSSSS,USER=99.

The I.4 UCSD PASCAL system is available for use. The UCSD PASCAL consists of a file handler, an editor, a compiler, and some utility functions. FLOPPY is the name of the user file and USER is the number of the DISK USER the file is written on. The emulator has the command set RUN, STEP, ECHO, NOECHO, and <@>. The ECHO, NOECHO commands act like the UYK-7/UYK-20 print ON/OFF commands in that ECHO turns the printer on and NOECHO turns it off. A more detailed description of the UCSD PASCAL emulator and its usage is described in a user's guide for UCSDP PASCAL. This is available with other QM-1 manuals in the NSWC QM-1 room.

V.32 PATCH-CALC

This command is intended to assist the SIMPL-Q programmer who is patching the machine code. Capabilities include calculating the PRU (128 18-bit words) number, an immediate operand, an indirect link, a stack address operand, a PC-relative address operand, and the 36-bit integer form of a given entry name as well as the reverse. The subcommands to accomplish this are as follows:

1. <@> - Leave Patch Calculator.

2. PRU-ADDRESS,ADDRESS=777777 - To aid patching on disk, this calculates the PRU number (decimal and octal) and the word in the PRU (octal) for the given address.

3. IMMEDIATE,VALUE=FFFFFF,BASE=!B!O!D!X! - This calculates an immediate operand, which may be specified in binary, octal, decimal, or hexadecimal.

4. INDIRECT-LINK,AT=777777,TO POINT TO=777777 - An indirect link to be stored at the first address and to point to the second address is calculated.

5. STACK-ADDRESS,ADDRESS=777777,INDIRECT=!YES!NO!,HALFWORD=!YES!NO! - Calculates a stack address operand.

6. PC-REL-ADDRESS,AT=777777,ADDRESS=777777,INDIRECT=!YES!NO!, HALFWORD=!YES!NO! - Calculates a PC-relative address operand to be stored at the first address and to point to the second address.

7. CODE-NAME,NAME=ANNNNN,TYPE=!PROC!DATA! - Calculates the 36-bit integer form of the given entry name.

8. DECODE-NAME,VALUE=777777777777 - Calculates the symbolic name from the 36-bit integer.

Examples follow:

```
!!PRU-ADDRESS,ADDRESS=200.
      WORD 000 OF PRU 0001

!!IMMEDIATE,VALUE=22,BASE=0.
      000026

!!INDIRECT-LINK,AT=0,TO POINT TO=0.
      000000

!!STACK-ADDRESS,ADDRESS=333,INDIRECT=NO,HALFWORD=NO.
      015551
!!STACK-ADDRESS,ADDRESS=1,INDIRECT=NO,HALFWORD=YES.
      000055

!!PC-REL-ADDRESS,AT=0,ADDRESS=0,INDIRECT=NO,HALFWORD=NO.
      777741

!!CODE-NAME,NAME=$$$$$$,TYPE=PROC.
      333333 333333

!!DECODE-NAME,VALUE=333333333333.
      $$$$$$      PROC
```

V.33 PATH HANDLER LANGUAGE

PHL is a tool for testing software which uses the CPX path handlers. PHL was originally constructed to debug the path handlers, but it can also debug other software which uses the paths.

PHL has commands which start path I/O, wait for I/O completion, compare buffers, and inspect TIOCBs. PHL can simulate the path operations of a processor on one side of the path so the software on the other side can be tested.

The major internal structures of the PHL processor are ten buffers (0-9) and ten read-only data patterns (0-9). The buffers are used for all path I/O, and the patterns can be copied into the buffers or compared against the buffers. All patterns and buffers are 100 octal words long. PHL also makes eight ECBs (1-8) available for path I/O. Should an error occur during PHL processing, PHL aborts by printing and displaying an appropriate error message.

PHL is invoked by this template on CMDSYS:

PHL., SENDER ID = 7, EXEC FILE = SSSSSSSSSS.

where

SENDER ID - an octal digit inserted into the sender ID field of all TIOCBs when an SIO is issued by PHL.

EXEC FILE - a file of PHL commands to be executed. If no file is specified (default), commands are read from the keyboard.

PHL supports these commands.

1. SIO-NTDS,ECB=9,COW=77,PATH/SUBCH=777,BUF=9,WCR=77,C=1,CW=99,XW=99 - Builds an NTDS TIOCB and issues SIO. Path/Subch parameter requires a three-octal-digit response: first two are path number. Third is subchannel.

2. SIO-UYK-7,ECB=9,COW=77,PATH/SUBCH=777,DO=777777,D1=777777, D2=777777,D3=777777 - Builds a UYK-7 TIOCB (DO-D3 are four words of data to go into BAR,WCR,ISR,DSR, respectively) and issues SIO.

3. TIOCB,ECB=9 - Displays the TIOCB.

4. WAIT,ECB=9,FOR TROUBLE=77 - Waits for ECB complete, then inspects the trouble response. If the error in the TIOCB does not match what was expected, the PHL program is aborted.

5. CHECK,BUF=9,FOR PATTERN=9,# WORDS=77 - Compares the buffer with the pattern for the specified word count. A mismatch aborts the PHL program.

6. COPY,TO BUF=9,PATTERN=9,# WORDS=77 - Copies the specified number of words from a pattern into a buffer.

7. DUMP,BUF=9,WORD COUNT=77 - Dumps the contents of the specified buffer to the printer.

8. OPTIONS,DISPLAY=!PRINTER!CRT!BOTH! - Selects the destination of PHL output.

9. SET-DATA=9,WORD=77,NEW CONTENTS=777777 - Sets a particular word in a buffer to a desired value.

10. /@ - Terminates PHL processing and returns to normal system operation.

11. HIO,PATH/SUBCH=777 - Halt I/O on the given path and subchannel - This command is not currently implemented because the TCP overlay for HIO has not been delivered.

7.34 PRINT

This command prints a file on the printer. The file may be either a disk file or a deck of cards in the card reader. If a disk file is to be listed, it must be a file of card images, such as produced by COPY or MACRO. The syntax for this command is:

PRINT,FILE=SSSSSSSSSS,SCAN=AAA.

Defaults are *CDR,OFF. If SCAN=ON, then lines that begin with eject, skip, or space directives are not printed--instead the compiler directive is executed.

V.35 PRINT-SPOOL

File used as print-spool for virtual printer will be printed. The syntax for this command is:

PRINT-SPOOL,FILE=SSSSSSSSSS. Default is 200UT:CDR.

V.36 PRU-MOD

The PRU-MOD command is used to modify the values stored on a disk file. The user specifies the file, the starting location to change in the file as indicated by the PRU number and word number, the new value, and the number of consecutive locations to change. The syntax for this command is:

PRU-MOD,FILE=SSSSSSSSSS,PRU #=777777,WORD #=777,VALUE=777777,
COUNT=777.

The defaults are QM1,0,0,0,0. All numbers entered are in octal, and if the defaults are entered, no change will occur.

V.37 PRUDMP

This command allows examination of a specified disk file by PRUS (128 18-bit words). A number of options are available to allow specification of desired characteristics when dumping a file.

The syntax for this command is:

PRUDMP,FILE=SSSSSSSSSS,SKIP=9999,DUMP=9999,TO=SAAA,TRANS=S,FMT=A99.

The FILE parameter specifies the disk file to be dumped (default is QM1). SKIP defines the number of PRUs to skip (starting with zero) before initiating output, while DUMP defines the actual number of PRUs to be dumped (defaults are 0 and 9999, respectively). The TO parameter specifies the physical device ('*LPT' or '*CRT') that the output will be displayed on (default is *LPT). When outputting to the CRT, one PRU is displayed on the screen and then the user is prompted to depress the 'RETURN' key to display the next PRU or the 'ESC' (escape) key to discontinue the dump. If 'BOTH' is entered for the TO parameter, the file will be dumped on the CRT and line printer.

The TRANS and FMT parameters allow specification of the display characteristics that the output is to have. TRANS specifies one of three possible bit translations for converting data into display code. The characters generated by the selected bit translation will appear to the right of the data words on a line of output. The translations are (1) 'E' which converts the low order 8 bits of each 18-bit word into ASCII, (2) '6' which converts every 6 bits of a word into a modified EXTBCD code (this code is compatible with the formatting of names in module headers), and (3) '8' which converts the low

8 bits of every 9 bits in a word to ASCII (default is 8). FMT allows specification of half (18-bit) or full (36-bit) word display with a desired conversion of octal, decimal, or hexadecimal. There are six options in specifying the format (O18,O36,D18,D36,X18, and X36, default is O36).

NOTE: This utility will suppress the printing of a PRU if every line in that PRU is equal to the last line of the previous PRU. A message is output notifying the user that a PRU has been suppressed.

V.38 QCONTROL

This command enables and disables the EASY debugging facility. Initially, it is enabled. When enabled, striking control-Q will invoke the debugger; when disabled, control-Q is treated as a normal input character.

The syntax for this command is:

QCONTROL,MODE=AAA. (ON/OFF)

Default is OFF. This command is "locked" (i.e., it may only be used in privileged mode of operation).

V.39 RDTAPE

This command will read a tape for use with the CDC 6000 and will list it on the printer. The syntax for this command is:

RDTAPE,UNIT=17. (Default is 0).

V.40 REWIND

This command will rewind tapes or disk files. The syntax for this command is:

REWIND,A=SSSSSSSSSSS,B=SSSSSSSSSSS,C=SSSSSSSSSSS,
D=SSSSSSSSSSS,E=SSSSSSSSSSS.

Default is 0,,,,.

V.41 SIMPLQ

This command is used to invoke the QM-1 SIMPL-Q compiler.

The syntax for this command is:

SIMPLQ,OPTIONS=AAAAAAA,INPUT=SSSSSSSSS,BINARY=SSSSSSSSS,
SCHEMA=SSSS3SSSSS.

OPTIONS - S - Print source listing
M - Macro pass
I - Use indirect links

L - Print EASY code produced
F - Generate attribute and cross-reference listing

These are the most useful options. A complete list of options may be found in the SIMPL-Q Reference Manual. The options may be combined in any meaningful combination. The default is SFMI.

INPUT - name of the input file. The default is *CDR, the card reader. If input is a disk file, it must be in card-image format [i.e., READF(STRING)].

BINARY - name of the output file. The default is QM1.

SCHEMA - name of the output file for symbolic debugging. The default is SCHEMA. The symbol table is written to the schema file which also forces the cross-reference option. The schema file can be incorporated into a schema library by the use of EDITLIB. A $ for file name will override the SCHEMA option.

V.42 SKIPBACKWARD

This command allows the user to skip backwards a specified number of files on a tape. The syntax for this command is:

SKIPBACKWARD,UNIT=17,NUM OF FILES=999,PARITY=1.

SKIPBACKWARD reads the last file mark skipped - for example skipping backward X files when positioned in file number n will position the tape at the beginning of file number

n if X = 1
n - 1 if X = 2, etc.

If X .GE. n, then of course the tape will be positioned at load point (unless the first record on the tape is a file mark in which case the tape will be positioned immediately after the file mark). Defaults are 0,1,1.

V.43 SKIPFORWARD

This command allows the user to skip forward a specified number of files on a tape. The syntax for this command is:

SKIPFORWARD,UNIT=17,NUM OF FILES=999.

Defaults are 0,1.

V.44 SM-TO-DISK

This command copies CDC 6700 produced 7-track tape to a QM-1 disk file. The tape is produced by the SMTAPE program. The syntax for this command is:

SM-TO-DISK,UNIT=17,FILE #=99,TO FILE=SSSSSSSSSS,REWIND=AAAAAA.

[illegible] tape unit number. Default is 0.
[illegible] number of files to space forward before copying begins. Default is 0.
[illegible] disk file name. Default is QMI.
REWIND = BOTH (rewind before file searching begins)
AFTER (rewind after copy)
BOTH (rewind before and after)
NONE (no rewind). Default is NONE.

[illegible] TO DISK

[illegible] command copies a source file from tape to disk. The tape format read [illegible] 80 (80 column) card images per block, ASCII encoded. Tapes in this [illegible] are generated on the 6700 by use of the CTOTAPE or STOTAPE procedures on [illegible] SSC.

[illegible] for this command is:

SOURCE-TO-DISK,TAPE UNIT=17,DISK FILE=SSSSSSSSS.

where:

TAPE UNIT - tape unit where tape is mounted. Default is 1.
DISK FILE - disk file where source is to be stored. Default is *DECK.

Two modes are supported. If a disk file name is specified, one file is [illegible] from tape to that disk file. If "*DECK" is used as the file name, the [illegible] file is scanned for *DECK cards (CDC UPDATE format). Each occurrence of a [illegible] card in the source file will make a card with the appropriate NOVA command [illegible] create space on a disk and assign that deck name to it. To be compatible with [illegible] NOVA file naming conventions, all "$" characters found in a deck name are changed [illegible] "S".

[illegible] "*DECK" mode is useful for transferring an UPDATE "SOURCE" file to the [illegible] on a "QM-1-file-per-deck" basis.

[illegible] HP

[illegible] HP command is used to present a histogram showing the percentage of [illegible] that the value of a specified location falls within a user-defined bin. [illegible] of this command is:

HP,EXPR=SSSSSSSSSSSSSSSSSS,MIN=777777,MAX=777777,
BINSIZE=777777,MASK=777777.

[illegible] are "HP,0,777777,10000,777777. The user specifies the minimum, maximum [illegible] bin size. The histogram will illustrate the percentage of samples below [illegible], equal or greater than minimum but less than minimum + bin size, ..., [illegible] than maximum. Figure L-8 of the UYK-7 CPU section illustrates a sample [illegible].

The sample is taken once every 100 ms following the SPY entry and is terminated with the histogram printout by entering <@>. The address sampled can be in the form MS (address) for a main store address or CS (address) for a control store address. All stack identifiers are also supported, (TS, BS,PC, etc.) as well as arithmetic functions +, -, *, and / and logical operators .A., .O., .X., .NA., .RL., .LC., and .LL.. The parameter MASK allows a portion of the sampled value to be masked off for use in the plot, via a logical AND(.A.).

SPY is useful for studying timing, such as seeing how other procedures are executed. For example, on the UYK-7 if ISWTCH is a procedure switch which branches to 10 different procedures based on a value of 0-9, then by making MIN=0, MAX=10, BINSIZE=1 the user could study the percentage of times each procedure was called.

Two general-purpose routines were developed for SPY: a scheduler, and a procedure to handle clock interrupts. SCHED allows the user to add a procedure to the list of procedures to be called when a clock interrupt occurs. The user may specify the number of interrupts to occur before the procedure is executed. The procedure which handles the clock interrupts CLKTNT determines which procedures are to be executed at the time of the clock interrupt.

V.47 SPY-PROC

The SPY-PROC command allows the user to study the time each procedure/ function takes. A table is built for up to the specified number of most recently used procedures. When an <@> is entered, the table indicating the percentage of CPU time and the names of the procedures will be printed. The syntax of this command is:

SPY-PROC,# PROCS=9999. (Default is 400).

V.48 TERMSIM

TERMSIM,LDID=77,BAUD=!300,!600,!1200!.

This command is used to initialize the QM-1 as an intelligent terminal. The TERMSIM command set allows the user to log in at a remote site (e.g., CDC 6700). Additionally, the user may wish to transfer files to/from the QM-1 before/after file editing, etc.

A modem must first be connected to the selected LDID. This should be connected through a null connection on the CROSSPATCH panel. For example, if the 300 baud rate modem is used and connected to LDID 13, then to initiate TERMSIM the user would type:

TERMSIM,LDID=13,BAUD=300.

The system will then ask for an escape character. The escape character allows the user to gain access to the TERMSIM commands following initiation of SIM-MODE. A character which is not used in communication with the remote computer should be chosen. An example would be CNTRL-A. The ESCAPE-RESET command can be used to modify the SIM-MODE escape character.

Prior to turning on the SIM-MODE, a file of commands to the remote computer may be built. The command FROM-FILE sets up the desired file. The syntax is FROM-FILE,=SSSSSSSSSS. To build the file, the user types EDIT which gives the user the ability to create and modify files. The EDIT command calls up the CRT editor which is identical to the editor called by the system command EDIT. See the discussion of the system command EDIT for a detailed description of EDIT.

To access the CDC 6700, the user should place the QM-1 in SIM-MODE and dial 663-6611 (300 baud) or 663-6641 (1200 baud). The QM-1 user can then use the QM-1 as a terminal. The ESCAPE character will return the user to the TERMSIM command template without terminating communications with the remote computer. To perform additional terminal communications the user types SIM-MODE or <CR>.

If the remote computer requires half-duplex rather than full-duplex, then the command ECHO is used to turn on the echo capability.

ECHO = !ON!OFF!.

When information from the 6700 (remote) is to be transferred to the QM-1, the TO-FILE command is used. This will transfer all data displayed at the CRT to the file. The syntax is TO-FILE,=SSSSSSSSSS.

The status of the TERMSIM can be displayed by using the STATUS command. A typical display is as follows:

```
QM1 TERMINAL SIMULATOR

STATUS:
PORT LDID = 13          BAUD RATE = 300
ESCAPE CHAR = ^A        ECHO = OFF(FULL DUPLEX)
TOFILE = NOT ACTIVE     FROMFILE = NOT ACTIVE
```

Two file copy routines are available in TERMSIM. These two commands are used to transfer NOVA files by converting from NOVA to SIMPL-Q format. The copies are:

1. COPYNS,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=!END!ZEROS!ETX!EOE! - Copy a NOVA file to a SIMPL-Q file.

2. COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=!ZEROS!ETX! - Copy a SIMPL-Q file to a NOVA file.

To terminate TERMSIM and return to the other system commands, the user can either enter QUIT or <@> which terminates the terminal simulator mode. For these commands to work, the TERMSIM template must first be activated. Thus, if SIM-MODE was entered, it must be terminated by the appropriate escape character. TERMSIM is then terminated by the QUIT or <@>.

V.49 TEST

The EASY server TEST allows execution of programs which are not part of the EASY system library.

The syntax for the TEST command is:

TEST,ENTRY=ANNNNN,FILE=SSSSSSSSSS.

where

ENTRY - entry point at which execution is to begin
FILE - file name of QM-1 disk file on which program resides

Note that the entry point which is invoked must not have any associated parameters. Therefore, if one does wish to test a program with parameters, a driver routine must be written which is invoked by the EASY TEST service and which, in turn, invokes the program to be tested.

V.50 TESTCF

This command tests a command file. It simply calls MASTER using the specified command file.

The syntax for this command is:

TESTCF,COMMAND FILE=SSSSSSSSSS.

Default file name is CMDTMP.

V.51 TIME

This command allows the user to reset the clock and is identical to the time entry during initialization. This command, as part of the system command template, allows the user to modify the time without the need of reinitializing.

V.52 TRANS

This command will transfer a source file to a 9-track tape for the CDC 6000. The syntax for this command is:

TRANS,UNIT=17,FILE=SSSSSSSSSS. (Defaults: 0,CDR).

V.53 UNLOAD

This command unloads tapes. The syntax for this command is:

UNLOAD,UNITA=1,UNITB=1.

Default is 0,.

V.54 UNLOCK

This command is used to go from "locked" to "unlocked" mode of operation. This command verifies that the user is privileged (i.e., knows the system password). If the password offered is correct, then the system is set into

privileged mode. The system header is rewritten to reflect the mode change ('EASY SYSTEM' replaced by 'UNLOCKED') and the system flag "UNLOCKED" is set to true. Now the user may execute commands previously denied (i.e., those designated as "locked").

The syntax for this command is:

UNLOCK,PASSWORD=SSSSSSSSSS.

V.55 UYK-7

The UYK-7 command is now implemented from CPX by using EFUYK7 or EFIOC7. This command should not be used at the command system level.

V.56 200UT

The CDC 200UT is a medium speed (i.e., 9600 baud) RJE terminal for use with CDC 6000-7000 series computers. A 200UT consists of an interactive (CRT) terminal, a card reader, and a line printer.

The 200UT emulator consists of four separate TCP tasks consuming 7,000 (octal) locations in control store and roughly 30,000 locations of main store (constants and buffers). The four concurrent tasks communicate via buffers and flags in main store. The tasks are:

1. Communications line controller (protocol handler),

2. Console terminal emulator (formatting and control),

3. Line printer driver, and

4. Card reader driver.

The special-purpose control program (UT200X) consists of roughly 100 SIMPL-Q statements. This interface supports the following commands, the loading and unloading (resetting of control store) of the emulator, and simulates the keyboard interface provided by PROD.

The EASY command 200UT invokes the 200UT emulator. The emulator is loaded into low CS and MS from binary modules produced by EASY command MODULATE. The contents of the CS used by the emulation are saved and restored from an array allocated in MS.

1. Loading and initializing the 200UT emulator -

a. At the patch panel, connect the QM-1 SYNC (QM-1 5 or QM-1 12) data port (upper position) to the 200-UT data port (middle position).

b. Type "200UT". Keyboard control is passed to the 200UT in "Entry" mode.

c. Dial 663-6691 and on tone, push red button. The system will respond with "ENTER."

d. Type "LOGIN." - The system will respond with "ENTER USER NAME-". User must enter appropriate CDC assigned user number followed by <CR>. The system will respond with "ENTER PASSWORD-". User replies with appropriate CDC charge code followed by <CR>. The system will respond with "COMMAND-" and is now ready to accept user commands.

2. Entering 200UT special keys - The 200UT emulator is always in "NORMAL" mode and "LINE" mode. The "SEND" key of the 200UT is replaced by the "RETURN" key of the CRT. The two remaining functions are "INT" and "LOAD". The 200UT emulator supports these special keys plus additional commands under "Command" mode and "Service" mode.

3. Entering the Command mode: A single escape brings up the 200UT emulation "Command" mode as defined in the Nanodata 200UT User's Guide. All function commands are terminated by (RETURN).

a. 200UT standard remote batch functions. (Note that ordinarily a 200UT only supports the card reader and printer as I/O devices).

(1) L - Load cards. Start a card reading sequence.

(2) I - Interrupt loading. Terminate card reading.

(3) OK - Acknowledge printer problems.

(4) P - Print local. Copy card to print until hopper empty or "I" or "IP".

(5) IP - Interrupt printing. Terminate printing, the last buffer is lost.

b. QM-1 extended functions (enhanced to support disk files).

(1) 9 - Set reader translate for 029 keypunch.

(2) 6 - Set reader translate for 026 keypunch.

(3) LD - Set Disk mode for the reader, rewinding the disk file.

(4) LD* - Same as LD but no rewind.

(5) PD - Set Disk mode for printing, rewinding the disk file.

(6) PD* - Set Disk mode for printing without rewinding the file.

(7) LU - Unset Disk mode for reading.

(8) PU - Unset Disk mode for printing.

NOTE: To load from disk, enter "LD", then "L". "I" interrupts loading, another "L" continues loading where left off [data may be lost]. To print disk file, enter "LD", then "P". To load cards to disk, enter ["LU" if necessary], "PD", "P". To copy disk files, enter "LD", "PD", "P". "IP" interrupts printing, another "P" continues printing with modes left all the same.

4. Entering the Service mode - Two escapes return control of the CRT to MASTER supporting the 200UT command file. (Line communications are maintained, and messages from the CDC while in this mode are discarded.) The following commands are supported:

a. COPY - copy a disk file (or cards) to another disk file.

Syntax: COPY,FROM=SSSSSSSSSS,TO=SSSSSSSSSS.

Defaults: *CDR,CDR. CARD READER TO TEMP DISK FILE.

b. COPYNS - copy a NOVA file to a SIMPL-Q file.

Syntax: COPYNS,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=!END!ZEROS!ETX!EOE!.

Defaults: 200UT:LPT,CDR,ETX.

c. COPYSN - copy a SIMPL-Q file to a NOVA file.

Syntax: COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=!ZEROS!ETX!.

Defaults: CDR,200UT:CDR,ETX.

d. DIRECTORY - select file directory search order.

Syntax: DIRECTORY,SEARCH 1ST=19,2ND=19,3RD=19,REQ DIR=,REQ DIR=,REQ DIR=.

Defaults: ,,,03,04,00.

e. EXEC - execute a file of commands.

Syntax: EXEC,FILE=SSSSSSSSSS.

Defaults: *CDR.

f. IMPORT - import binary file shipped in from CDC 6700.

Syntax: IMPORT,TO FILE=SSSSSSSSSS.

Defaults: QM1.

g. KILL - kill the 200UT emulator and return control to EASY.

Syntax: KILL. (This command does not allow the user to wait for LOGOUT information.)

h. LOGIN - transfers a default login sequence to the CDC 6700.

Syntax: LOGIN,PROGRAMMER NAME=NNNNNNNNNN,
ACCOUNT=NNNNNNNNNN.

Defaults: N68PERRY,99KK33.

i. LOGOUT - logout on the 200UT and return to EASY.

Syntax: LOGOUT. (LOGOUT information given from the CDC 6700)

j. <@>, <CR>, <ESC> - escape from the 200UT "SERVICE" mode and return control to 200UT "ENTRY" mode.

NOTE: The 200UT emulator has numerous diagnostic stops enabled if "F" switch #4 is turned on. If loading the version of the 200UT that is on track 3 of the NOVA System cart, you must set the F switches to 0 after pressing the start button. (It is a good idea to always zero the sense switches unless you intend to debug an emulator, TCP, TBP, 200UT, and other software products depend on the F switches to signal diagnostic halts.)

V.57 <@>

This command will cause an "EASY Recovery Deadstart" to occur when executed at the command level. The user can use the control key <@> to execute this command.

VI. EASY DEBUG FACILITY

The QM-1 programmer has available an interactive, low-level debugging facility when running under the EASY system. This facility will allow the user to inspect or change the contents of main store or control store. Memory locations are referenced by relative QM-1 addresses and the contents may be displayed, dumped, or specified in various formats. The debug facility may be accessed by simply pressing the CNTRL and Q keys simultaneously. This operation will suspend the currently executing EASY program except for higher level interrupt handlers such as I/O and fault processors. The debug operations are invoked via the MASTER command interpreter and the user is notified by a CRT message that the system is active.

The following is a list of debug commands and a brief explanation of how each operates. Some of the commands require a SCHEMA file in order to access cross-reference information. The SCHEMA files are produced by the SIMPL-Q compiler when the modules are being compiled.

Some of these commands will prompt for an address to be supplied. Addresses for these commands may be a standard address. If the user desires to specify an indirect address (and it is in SIMPL-Q format), the address must be proceeded by the letter I. The array displays are paged: for character arrays, 420 elements are displayed at once. Six elements are displayed for string arrays and 25 for integer arrays. The "+" and "-" commands are used to display other parts of the array. The first element to be displayed can be entered as a parameter to the ARRAY command.

1. <*> - Displays a short program space map which is a subset of MAPPROGSPACE discussed later.

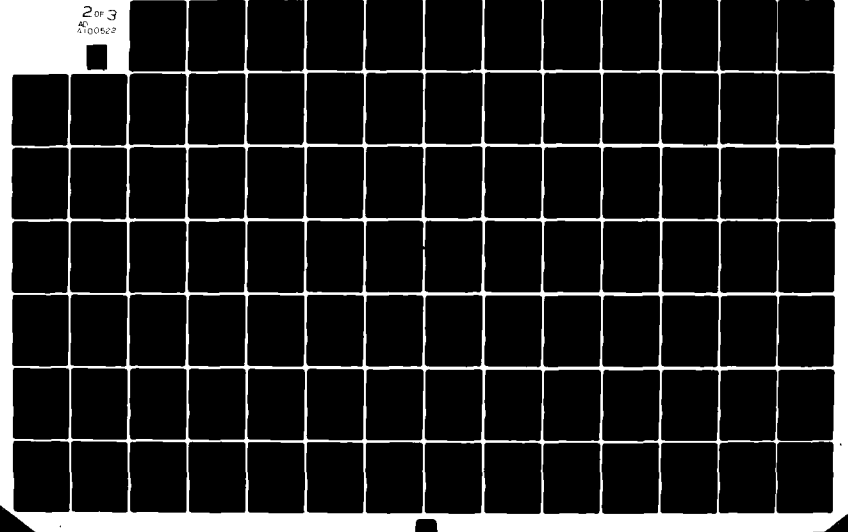
```
MODULE : PSMKRL (622536/000030)    0.0    01/31/80    21:58:15
MODULE : DEVSTY (622566/000160)    2.6    01/31/80    21:30:40
MODULE : CPXSVI (622746/006264)    2.6    01/31/80    21:22:40
MODULE : SCPX   (631232/036014)    2.6    01/31/80    22:16:28
```

2. <+> - Page display forward. For this example, the previously selected memory display was for location 1000.

```
!!+!.
  MAIN STORE
001100   000000  000000  000000  000000  000000  000000  000000  000000
001110   000000  000000  000000  000000  000000  000000  000000  000000
001120   000000  000000  000000  000000  000000  000000  000000  000000
001130   000000  000000  000000  000000  000000  000000  000000  000000
001140   000000  000000  000000  000000  000000  000000  000000  000000
001150   000000  000000  000000  000000  000000  000000  000000  000000
001160   000000  000000  000000  000000  000000  000000  000000  000000
001170   000000  000000  000000  000000  000000  000000  000000  000000
```

AD-A100 522 NAVAL SURFACE WEAPONS CENTER DAHLGREN VA F/G 9/2
EMULATION AID SYSTEM II (EASY II) SYSTEM PROGRAMMER'S GUIDE.(U)
MAR 81 C J NAPLES
UNCLASSIFIED NSWC/TR-81-98 NL

2 OF 3
AD A100522



3. <-> - Page display backward.

```
!!<->.
   MAIN STORE
  001000   000000  000000  000000  000000  000000  000000  000000  000000
  001010   000000  000000  000000  000000  000000  000000  000000  000000
  001020   000000  000000  000000  000000  000000  000000  000000  000000
  001030   000000  000000  000000  000000  000000  000000  000000  000000
  001040   000000  000000  000000  000000  000000  000000  000000  000000
  001050   000000  000000  000000  000000  000000  000000  000000  000000
  001060   000000  000000  000000  000000  000000  000000  000000  000000
  001070   000000  000000  000000  000000  000000  000000  000000  000000
```

4. <@> - Ends debugging task and returns to previous control level.

5. <CR> - Display last memory display again. See next command for example.

6. <LF> - Prints the current display. In this example, a <CR> had been entered to display memory 1000 to 1077. Then the <LF> was entered.

```
!!<CR>.
   MAIN STORE
  001000   000000  000000  000000  000000  000000  000000  000000  000000
  001010   000000  000000  000000  000000  000000  000000  000000  000000
  001020   000000  000000  000000  000000  000000  000000  000000  000000
  001030   000000  000000  000000  000000  000000  000000  000000  000000
  001040   000000  000000  000000  000000  000000  000000  000000  000000
  001050   000000  000000  000000  000000  000000  000000  000000  000000
  001060   000000  000000  000000  000000  000000  000000  000000  000000
  001070   000000  000000  000000  000000  000000  000000  000000  000000
!!<LF>.
```

7. <PERIOD> - Display short map of stack.

```
!!<PERIOD>.
   NAME    BS(ABS)    LINE #     STR     MODULE   VERSION     DATE        TIME
  STKMAP    416500    ?????    000011   STKMAP      2.6     01/31/80   22:03:33
  CMDIT     415134      248    000011   MASTER      2.6     03/06/80   08:48:58
  MASTER    415014      210    000011   MASTER      2.6     03/06/80   08:48:58
  DBUGR     414732       95    000011   DBUGR       2.6     01/31/80   21:27:09
  DBUGER    414726       36    000011   $FAULT      2.6     01/31/80   22:21:05
  $KBCHA    414704       86    000000   $KYBRD      2.6     01/31/80   22:28:58
  FIRSTC    414650      644    000000   MASTER      2.6     03/06/80   08:48:58
  KEYPRO    411300      282    000000   MASTER      2.6     03/06/80   08:48:58
  CMDIT     407734      231    000000   MASTER      2.6     03/06/80   08:48:58
  MASTER    407614      210    000000   MASTER      2.6     03/06/80   08:48:58
  $CPX      407600      191    000000   $CPX        2.6     01/31/80   22:16:28
  CMDIT     406234      248    000000   MASTER      2.6     03/06/80   08:48:58
  MASTER    406114      210    000000   MASTER      2.6     03/06/80   08:48:58
  $YSTEM    406104      107    000000   $YSTEM      2.6     01/31/80   22:39:11
  DDSTRT    406100        1    000000   DDSTRT      0.0     01/00/00   00:00:00
```

8. ARRAY,ADDRESS=<array address>,FIRST ELEMENT=<first element number> - Display SIMPL-Q array.

```
!!ARRAY,ADDRESS=130,FIRST ELEMENT=0.
   INTEGER
(0)
              0              0              0              0              0
  000000 000000  000000 000000  000000 000000  000000 000000  000000 000000
(5)
              0              0              0              0              0
  000000 000000  000000 000000  000000 000000  000000 000000  000000 000000
(10)
              0              0              0              0              0
  000000 000000  C00000 000000  000000 000000  000000 000000  000000 000000
(15)
              0              0              0              0              0
  000000 000000  000000 000000  000000 000000  000000 000000  000000 000000
(20)
              0              0              0              0              0
  000000 000000  000000 000000  000000 000000  000000 000000  000000 000000
```

9. BASE,ADDRESS/MODULE=<address or /module name or +proc name> - Establishes the base address for the main store displays. This command is valid only for EASY machines. The response indicates the starting address of the module. When a '+' is the first character given, the following characters are taken to be the name of a procedure/function on the stack. The base address will be set to the bottom of the highest stack frame for the named procedure.

```
!!BASE,ADDRESS/MODULE=/$1004.
MODULE $1004 STARTS AT 073154
```

10. C,ADR=<address>,FORMAT=<format indicator> - Displays control store starting at specified address in the format requested. The format indicator is interpreted in the following manner:

O = Octal
D = Decimal
X = Hexadecimal
F = Full word - may be appended to each of the three bases (O, D, and X).

```
!!C,ADR=132,FORMAT=0.
   CONTROL STORE
000132   000000  000000  000000  000000  000000  000000  000000  000000
000142   000000  000000  000000  000000  000000  000000  000000  000220
000152   000017  000000  000200  777777  000370  000360  000350  000340
000162   000330  000320  000310  000300  000270  000260  000250  000240
000172   000230  000000  000210  000000  000220  000173  000510  000000
000202   200717  777777  000000  000000  066252  077757  000004  000000
000212   000000  000000  000000  000000  000000  000000  000000  000003
000222   100420  020040  073404  000034  000000  000000  000004  000000
```

11. CALCULATE,EXPRESSION=<arithmetic expression>,BASE=<base> - Calculates the value of an expression in the base specified. No input will cause a repeated prompt for input. To exit this mode, enter END.

```
!!CALCULATE,EXPRESSION=4*5,BASE=8.
INPUT EXPR=4*5
VALUE =000000000024
!!END.
```

12. CHANGE,SYMBOL=<symbol name>,IN PROC=<procedure name> - SCHEMA will be used to determine location of symbol in the procedure. Contents of the variable (scalar or array) will be changed to the specified value. If the symbol is an array, the user will be prompted for an index (only one element is changed).

For characters and integers, any constant accepted by the compiler may be input for a new value. NOTE: Do not enter the smallest negative integer (i.e., 0'4211').

For strings, any keystroke is accepted as part of the string, except a single quote, which terminates the string. Single quotes may be entered into the string by typing two successive single quotes (the cursor twitches but the second quote is not displayed). No more characters can be typed than the maximum length of the string being changed. Current limit of max length is 66 characters. If a mistake is made while entering the new string, type a single quote to terminate the string and then hit the escape key.

If SCHEMA is not available, the following printout will occur.

```
!!CHANGE,SYMBOL=JOE,IN PROC=ASD.
NO SCHEMA
```

13. CHANGECS,ADDRESS=<starting address>,VALUE=<new value>,REP=<number of times new value is to occur> - Changes the value of the specified location to the value. Repeat option allows up to 999 locations to be changed to the same value.

```
!CHANGECS,ADDRESS=130,VALUE=0,REP=1.
   CONTROL STORE
000130   000000  000000  000000  000000  000000  000000  000000  000000
000140   000000  000000  000000  000000  000000  000000  000000  000000
000150   000000  000220  000017  000000  000200  777777  000370  000360
000160   000350  000340  000330  000320  000310  000300  000270  000260
000170   000250  000240  000230  000000  000210  000000  000220  000173
000200   000530  000000  200717  777777  000000  000000  066252  077757
000210   000004  000000  000000  000000  000000  000000  000000  000000
000220   000000  000003  100420  020040  073306  000034  000000  000000
```

14. CHANGECSM,ADDRESS=<starting address>,VALS=<multiple values> - Changes the contents of control store, starting at the specified address for as many locations as values are specified. The values are separated by a /.

```
!CHANGECSM,ADR=130,VALS=0/77777/0/77777/0/77.
   CONTROL STORE
000130   000000  077777  000000  077777  000000  000077  000000  000000
000140   000000  000000  000000  000000  000000  000000  000000  000000
000150   000000  000220  000017  000000  000200  777777  000370  000360
000160   000350  000340  000330  000320  000310  000300  000270  000260
000170   000250  000240  000230  000009  000210  000000  000220  000173
000200   000530  000000  200717  777777  000000  000000  066252  077757
000210   000004  000000  000000  000000  000000  000000  000000  000000
000220   000000  000003  100420  020040  073306  000034  000000  000000
```

15. CHANGEMS,ADDRESS=<starting address>,VALUE=<new value>, STEP=number of times the new value is to occur> - Changes main store in the same manner as item 13 above, modifies control store.

16. CHANGEMSM,ADDRESS=<starting address>,VALS=<multiple values> - Changes the contents of main store, starting at the specified address for as many locations as values are specified. The values are separated by a /.

```
!CHANGEMSM,ADR=200304,VALS=1/2.
   MAIN STORE
200304   000001  000002  000000  000000  000000  000000  000000  000000
200314   000000  000000  000000  000000  000000  000000  000000  000000
200324   000000  000000  000000  000000  000000  000000  000000  000000
200334   000000  000000  000000  000000  000000  000000  000000  000000
200344   000000  000000  000000  000000  000000  000000  000000  000000
200354   000000  000000  000000  000000  000000  000000  000000  000000
200364   000000  000000  000000  000000  000000  000000  000000  000000
200374   000000  000000  000000  000000  000000  000000  000000  000000
                                                                      ;
```

17. CHARACTER,ADDRESS=<address> - Displays SIMPL-Q character variable at the specified address. This character is displayed as an octal value.

```
!!CHARACTER,ADDRESS=203504
    012
```

18. CODE,MODULE=<module name>,LINE=<line number> - Displays EASY machine code in symbolic format. This command is not yet implemented. Entering the command will result in the following:

```
!CODE,MODULE=DDSTRT,LINE=1.
COMMAND NOT YET IMPLEMENTED!
```

19. CODEPATCH,MODULE=<module name>,ADDRESS=<address>, CODE=<symbolic machine code> - Patches a module via symbolic machine code. This command is not yet implemented. Entering the command will result in the following:

```
!CODEPATCH,MODULE=*,ADDRESS=0,CODE=HALT.
COMMAND NOT YET IMPLEMENTED!
```

20. CONVERT,THIS=<base 'number'>,TO BASE=<base> - Converts a number from one base to another (decimal, binary, octal, or hexadecimal).

```
!!CONVERT,THIS=0'24',TO BASE=D.
0'24' CONVERTED TO D FORMAT = 20
```

21. DISPLAY,SYMBOL=<symbol name>,IN PROC= <procedure name> - Displays the value of the symbol. Requires SCHEMA to determine type and location. If no SCHEMA is available, the following display will result.

```
!DISPLAY,SYMBOL=IBUFF,IN PROC=$1004.
NO SCHEMA
```

22. DISPLAYCS,ADDRESS=<address>,FORMAT=<format> - Displays the contents of the specified address in the requested format (O, D, or X as defined in item 10 above).

```
!DISPLAYCS,ADDRESS=130,FORMAT=0.

   CONTROL STORE
000130   001066  000000  000000  000000  000000  000000  000000  000000
000140   000000  000000  000000  000000  000000  000000  000000  600020
000150   203311  000003  001411  000012  001067  007704  203333  000271
000160   002035  012164  000271  000000  000000  000000  000000  000000
000170   000601  000100  001113  000000  000000  000000  000000  000000
000200   000000  000000  000000  000000  000000  000000  000000  004673
000210   006626  006653  000000  000000  000000  000000  000000  000000
000220   000000  000000  000000  000000  000000  000000  000000  000000
```

23. DISPLAYECB,ECB NUM=<ECB number>,PRINT=<yes or no> - Displays the ECB (Event Control Block) specified and also prints if requested.

```
!DISPLAYECB,ECB NUM=0,PRINT=YES.
:YES:   1
OCTAL DUMP OF ECB NO 0
                ECW:  000550
       INITIAL REGS:  000000
      REG SAVE AREA:  200717
              BLINE:  777777
               BID1:  000000
               BID2:  000000
                BPS:  066252
                TPS:  077757
```

ECB 17_8 is used for error isolation, and it is dumped in a different format. Of particular note is the field ERROR NO. of ECB 17_8. The value of this field may be matched against the list of system error codes in Appendix B to determine why the fatal error occurred.

```
OCTAL DUMP OF ECB NO 15
    ECW: 000000000006
    ERROR NO: 000000001036
    BS AT ERR: 000000307272
    TS AT ERR: 000000307306
    ERR MESSAGE: 000000000000
                 : 000000000001
                 : 000000000000
                 : 000000472360
```

24. DISPLAYMS,ADDRESS=<address>,FORMAT=<format> - Displays main store memory starting at address specified in the requested format (O,OF,D,DF,X,XF). Format is the same as defined in item 10 above.

```
DISPLAYMS,ADDRESS=200304,FORMAT=O.

 MAIN STORE
200304  000001  000002  000000  000000  000000  000000  000000  000000
200314  000000  000000  000000  000000  000000  000000  000000  000000
200324  000000  000000  000000  000000  000000  000000  000000  000000
200334  000000  000000  000000  000000  000000  000000  000000  000000
200344  000000  000000  000000  000000  000000  000000  000000  000000
200354  000000  000000  000000  000000  000000  000000  000000  000000
200364  000000  000000  000000  000000  000000  000000  000000  000000
200374  000000  000000  000000  000000  000000  000000  000000  000000
```

25. EXEC,FILE=<file name> - This command causes a file of commands to be executed. An example of such a file with commands valid under the CPX level follows:

```
CREATE-MEMORY,1004,18,100000.
CREATE-PROCESS,4,1004,EA$Y.
DIRECTORIES,,,,17,03,00.
LIBRARIES,1004LIB.
<CR>
ASSIGN,DISK,4,10.
ASSIGN,PRTR,4,04.
ASSIGN,RDR,4,02.
CREATE-PATH,2,PERIPHERAL.
CONNECT,2,2,10.
CONNECT,2,4,0.
```

26. GO - Clears step or halt settings. Execution will automatically resume on leaving the debugger.

27. GOTO,LINE=<line number> - Changes the next statement to execute to the line specified. This command has not yet been implemented. If invoked, the following will occur.

```
!GOTO,LINE=*.
COMMAND NOT YET IMPLEMENTED!
```

28. HALT,AT LINE=<line number>,IN PROC=<procedure name> - Sets EASY breakpoint location.

```
!HALT,AT LINE=55,IN PROC=$1004.
```

29. INTEGER,ADDRESS=<address> - Displays SIMPL-Q integer variable at the specified address.

```
!!INTEGER,ADDRESS=130.
     7340034     000034 000002
```

30. LINE,NUMBER=<line number>,FOR N=<number of lines> - Displays N lines of source starting at specified line. This command has not been implemented and if invoked will produce the following display:

```
!LINE,NUMBER=1,FOR N=1.
COMMAND NOT YET IMPLEMENTED!
```

31. LOCK. Relocks the keyboard, making all locked commands unavailable.

32. M,ADR=<address>,FORMAT=<format> - Displays main store memory the same as item 24 above.

```
!M,ADR=130,FORMAT=0.
   MAIN STORE
  073304   000034  000002  000000  000000  000000  000000  000000  000000
  073314   000000  000000  000000  000000  000000  000000  000000  000000
  073324   000000  000000  000000  000000  000000  000000  000000  000000
  073334   000000  000000  000000  000000  000000  000000  000000  000000
  073344   000000  000000  000000  000000  000000  000000  000000  000000
  073354   000000  000000  000000  000000  000000  000000  000000  000000
  073364   000000  000000  000000  000000  000000  000000  000000  000000
  073374   000000  000000  777704  777555  516102  250606  000034  000002
```

33. MAPPROGSPACE,LEVEL=<level 1, 2, or 3>,PRINT=<yes or no>, STOP AT=<module name or number of modules to map> - Displays and prints a map of program space for the level requested. Level is interpreted as follows:

```
1 - module name, address, and length
2 - 1 + entry points and address
3 - 2 + external references
```

An example of level 1 follows:

```
!MAPPROGSPACE,1,YES,0.                    UNLOCKED    1.2.0   10/31/80 12:32:02

 MODULE : $PRINT (066252/002416)     2.6     01/31/80    22:32:57

 MODULE : CDR104 (070670/001224)     2.6     03/10/79    10:23:09

 MODULE : $YSTEM (072114/001040)     2.6     09/30/00    10:15:15

 MODULE : $1004  (073154/004544)     2.6     09/30/00    12:26:31

 MODULE : DDSTRT (077720/000040)     0.0     01/00/00    00:00:00
```

34. MAPSTACK,PRINT=<yes or no>,STOP AT=<module name or number of modules> - Displays the procedures on the stack. The stack is a list of procedures in the order they occurred.

```
!MAPSTACK,PRINT=YES,STOP AT=0.
   NAME  BS(ABS)    LINE #   STR     MODULE  VERSION     DATE        TIME
  GETBUF  000156        70  000002   $1004     2.6     09/30/00    12:26:31
  GETCHA  000120       116  000002   $1004     2.6     09/30/00    12:26:31
  MBOARD  000062       199  000002   $1004     2.6     09/30/00    12:26:31
  FIRSTI  000056       127  000002   $YSTEM    2.6     09/30/00    10:15:15
  MYSTEM  000036       112  000000   $YSTEM    2.6     09/30/00    10:15:15
  $YSTEM  000032        64  000000   $1004     2.6     09/30/00    12:26:31
  DDSTRT  000026         1  000000   DDSTRT    0.0     01/00/00    00:00:00
```

35. MBP,FUNC = { INI / SET / CLEAR / DIS }, 1st = 777777, 2nd = 777777, 3rd = 777777.

INI - Initialize the Micro Breakpoint File. Should be done once, after loading cart. Remainder of prompts ignored.

SET - Set micro breakpoints at address indicated by remaining prompts.

CLEAR - Clear micro breakpoints at address indicated by remaining prompts.

DIS - Display current status of micro breakpoints. Remainder of prompts ignored.

Controls the micro breakpoint capability and allows up to three breakpoints to be set by a single command.

```
!MBP,FUNC=SET,1ST=130,2ND=504,3RD=777.
ADDR  /INSTR      ADDR  /INSTR      ADDR  /INSTR      ADDR  /INSTR
------------      ------------      ------------      ------------
205034/774050     205034/774050     000130/000000     000504/000000
000777/000000           /                 /                 /
```

36. MBPPRO,ECB NO=<ECB number or T or D> - Proceed with emulator whose TASK START CONTROL BLOCK (TSCB) is located at indicated ECB #. The emulator must be halted at a micro breakpoint.

```
NOTE:  T   TRIDENT  TSCB
       D   DCP      TSCB
```

37. PATCH-CALC - Calculator for EASY machine code patches. This command is described in Section V.32.

38. PRINT,FILE=<file name>,SCAN=<on or off> - Prints card images on printer. If scan is selected as ON, the printer will obey /+EJECT+/ cards for top of page control and will include line numbers as would be generated by the compiler.

39. PRINTCS,FROM=<beginning address>,TO=<ending address>,FORMAT=<format indicator> - Prints control store memory from starting address through ending address in the format specified. Format is defined in item 10 above.

```
!PRINTCS,130,150,0.                          UNLOCKED    1.2.0   10/31/80 12:34:53

000130:   774006  000000  000000  000000  000000  000000  000000  000000
000140:  ALL ZEROS
000150:   000000  000220  000017  000000  000200  777777  000370  000360
000160:   000350  000340  000330  000320  000310  000300  000270  000260
000170:   000250  000240  000230  000000  000210  000000  000220  000173
000200:   000550  000000  200717  006700  336160  606400  066252  077757
000210:   000004  000000  000000  000000  000000  000000  000000  000000
000220:   000000  000003  100420  020040  073306  000034  000000  000000
```

40. PRINTMS,FROM=<beginning address>,TO=<ending address>, FORMAT=<format indicator> - Prints main store memory from starting address through ending address in the format specified. Format is defined in item 10 above.

```
!PRINTMS,200504,200604,0.                    UNLOCKED    1.2.0   10/31/80 12:35:20

273660:    023400  000050  003501  000122  724245  000043  000411  000054
273670:    000411  000307  776601  770641  776501  776441  776401  776341
273700:    765701  776241  776201  776141  776101  776041  776001  775741
273710:    775701  775641  775601  775541  775501  775441  775401  775341
273720:    775301  775241  775201  775141  775101  775041  775001  774741
273730:    774701  774641  774601  774541  774501  774441  774401  774341
273740:    774301  774241  774201  760101  760041  760001  757741  757701
273750:    757641  757601  757541  757501  757441  023500  000050  046741
273760:    023600  000020  720041  000007  000411  000052  000411  013001
273770:    024000  000050  007041  024400  000002  676601  700141  000415
274000:    000002  732521  000431  000515  000002  677001  677541  000615
274010:    000022  000631  000007  000711  024500  000002  675541  677101
274020:    001015  000002  731461  001031  001115  000002  675741  676501
274030:    001215  000022  001231  000007  001311  001672  001311  000007
274040:    001131  001415  000272  001431  001555  000272  001515  001515
274050:    000122  001555  001515  001515  000022  001515  000013  001515
```

41. Q-CONNECT,TO SON=<number of son> - Connects the debugger to any son (processor) under CPX. Specifying a son of zero connects the debugger to the parent.

```
!!Q-CONNECT,TO SON=4.
```

42. REGISTERS,ADR=<address>,TITLE FMT=<title format> - Displays local store registers. Title format (E,I,D,T) is interpreted as follows:

E - EASY titles
I - Interrupt machine titles

D - DCP titles
T - TRIDENT titles

Default is EASY registers, task format.

```
!REGISTERS,ADR=0,TITLE FMT = .
LOCAL STORE
    LS00     LS01     LS02     LS03     LS04     LS05     LS06     LS07
  000000   000000   000000   000001   000000   000000   000000   006626
    LS10     LS11     LS12     LS13     LS14     LS15     LS16     LS17
  006653   003022   001067   000000   000000   000000   000000   000000
    LS20     LS21     LS22     LS23     LS24     LS25     LS26    R.SYS
  000000   000000   000000   000001   003450   000000   000000   000000
     R.MX     R.IX     R.IY    R.MPC    R.ADR     FIDX     FMPC     FIST
  000000   000000   000000   000000   000000       00      000       00
```

43. REGISTERSET,ADR=<address>,REGISTER=<register number>,VALUE=<new value>,TITLE FMT=<title format> - Changes the contents of a register to the new value specified. Title format is the same as defined in item 42 above.

```
!REGISTERSET,ADR=1000,REGISTER=5,VALUE=707070,TITLE FMT=E.
LOCAL STORE
    ZERO    ZERO1      R.2      R.3       OR      OR1      ORA     ORA1
  000004   000003   200000   010020   000000   707070   000004   000003
     ORB     ORB1      MSA     MSA1       TS       BS       IR       LI
  201000   010020   000000   000000   000004   000003   202000   010020
    R.20     R.21     CTR1     CTR2      OPC      STK        U    R.SYS
  000000   000000   000004   000003   203000   010020   000000   000000
      PC     R.IX      STR    R.MPC    R.ADR     FIDX     FMPC     FIST
  000004   207000   000003   000004   000000       00       00       01
```

44. SC,ADR=<address>,VALS=<new values> - Sets control store memory, beginning at address specified, to the new value indicated. Multiple new values may be inserted by separating them with a /.

```
!SC,ADR=132,VALS=22222.
   CONTROL STORE
000132    022222  000000  000000  000000  000000  000000  000000  000000
000142    000000  000000  000000  000000  000000  000000  000000  000220
000152    000017  000000  000200  777777  000370  000360  000350  000340
000162    000330  000320  000310  000300  000270  000260  000250  000240
000172    000230  000000  000210  000000  000220  000173  000550  000000
000202    200717  006700  336160  606400  066252  077757  000004  000000
000212    000000  000000  000000  000000  000000  000000  000000  000003
000222    100420  020040  073306  000034  000000  000000  000004  000000
```

45. SCHEMA,FILE= <filename or filename$module or $module> - selects schema file for symbolic debug. FILE is interpreted as follows.

filename - indicates that the schema is contained on the named file.

filename$module - the named file is a schema library, and the schema for the named module is retrieved from library.

$module- the named module's schema is retrieved from the current library (which should be given in an earlier SCHEMA command using the second option for FILE).

Refer to SIMPL-Q (Section V.41) for additional information on a schema library.

```
!SCHEMA,FILE=SCHEMA.
   SCHEMA FOR MODULE $ MODULE $MODUL    09/30/00    10:06:38
```

46. SM,ADR=<address>,VALS=<new values> - Sets main store memory, beginning of address specified, to the new value indicated. Multiple new values may be inserted by separating them with a /.

```
!SM,ADR=200504,VALS=123456.
   MAIN STORE
  273660   123456   000050   003501   000122   724245   000043   000411   000054
  273670   000411   000307   776601   770641   776501   776441   776401   776341
  273700   765701   776241   776201   776141   776101   776041   776001   775741
  273710   775701   775641   775601   775541   775501   775441   775401   775341
  273720   775301   775241   775201   775141   775101   775041   775001   774741
  273730   774701   774641   774601   774541   774501   774441   774401   774341
  273740   774301   774241   774201   760101   760041   760001   757741   757701
  273750   757641   757601   757541   757501   757441   023500   000050   046741
```

47. SOURCE,FILE=<file name> - Changes source file to be used in debugging for commands such as LINE. This command is not yet implemented.

```
!SOURCE,FILE=SOURCE.
COMMAND NOT YET IMPLEMENTED!
```

48. STEP,PROC NAME=<procedure name> - Sets EASY stepping control for the specified procedure. The default (*) turns stepping off. Procedure name $$$ steps all procedures.

49. STRING,ADDRESS=<address> - Displays SIMPL-Q string variable.

50. TRACEBACK,LEVEL=<traceback level> - Invokes traceback. LEVEL is interpreted as follows:

1 - Procedure Calls

2 - 1 + Locals

3 - 2 + Globals

4 - 3 + Program

Procedure traceback consists of a history of procedure calls and a dump of the run-time stack frame associated with each call. The relative offset from the base of the stack frame provides a link between the traceback dump of the stack frame and the compiler cross-reference map enabling the user to find the value of locals at the time of crash.

The following is an example of a procedure traceback listing:

```
**********************************************************************************
*                                  P R O C E D U R E                             *
*                                  T R A C E B A C K                             *
**********************************************************************************

PROCNAME   $GETAV
MODULE     $SEQIO
LINE NUM   257
ID         777777470614(BASE 8)
PC         777777470631(BASE 8)
STR        000000000000(BASE 8)

LOCALS:

  REL      ABS      VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE
000000   307272   163032   035040   777742   472360   000000   000000   000000   473360
000010   307302   000000   472360   000000   000001


PROCNAME   $RDFS2
MODULE     $SEQIO
LINE NUM   229
ID         777777470370(BASE 8)
PC         777777470505(BASE 8)
STR        000000000000(BASE 8)

LOCALS:

  REL      ABS      VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE
000000   307224   163100   000071   777446   000071   000000   000200   000001   000002
000010   307234   000125   000123   000000   000200   000000   472360   000000   000000
000020   307244   000000   000601   000000   472360   000000   473360   000000   000200
000030   307254   000000   000001   000000   472356   000000   000000   466435   000071
000040   307264   000002   000071


PROCNAME   $RDFIA
MODULE     $SEQIO
LINE NUM   143
ID         777777467524(BASE 8)
PC         777777467577(BASE 8)
STR        000000000000(BASE 8)

LOCALS:

  REL      ABS      VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE    VALUE
000000   306660   163444   000016   777564   0000002  000000   000100   000000   000000
000010   306670   000000   000201   000000   000000   000000   000310   052056   046523
000020   360700   043460   035040   020040   020040   020040   020040   030015   005124
000030   306710   027115   051507   030472   020040   020040   020040   020040   020060
000040   360720   006412   052056   046523   043462   035040   020040   020040   020040
000050   306730   020040   030015   005124   027115   051507   054072   020040   020040
```

Global traceback consists of all global values associated with procedures listed in the history of procedure calls. The relative offset from the beginning of the module header provides a link between the global traceback dump and the compiler cross-reference listing enabling the user to find the value of globals at the time of crash.

The following is an example of global traceback listing.

```
*****************************************************************************
*                                  G L O B A L                              *
*                              T R A C E B A C K                            *
*****************************************************************************

MODULE    $SEQIO

GLOBALS:

 REL     ABS     VALUE   VALUE   VALUE   VALUE   VALUE   VALUE   VALUE   VALUE
000046  466456  000000  000400  000000  000000  000000  000000

MODULE    PRUDMP

GLOBALS:

 REL     ABS     VALUE   VALUE   VALUE   VALUE   VALUE   VALUE   VALUE   VALUE
000152  472324  000014  300002  042107  043111  046105  000000  000000  000003
000162  472334  472360  473360  472360  472360  000600  340007  000010  000000
000172  472344  472356  000000  000000  000000  000000  000000  000000  000000
000202  472354  000000  000000  001000  001000  020040  020040  020060  026124
000212  472364  027102  046116  045415  005040  020040  020040  020040  020112
000222  472374  051522  020040  020040  020100  050125  052103  006412  020040
000232  472404  020040  020040  020040  046104  040440  020040  020040  030454
000242  472414  052056  046523  043461  006412  020040  020040  020040  020040
000252  472424  046517  053132  046040  020040  030454  030415  005040  020040
000262  472434  020040  020040  020115  047526  046040  020040  020061  026060
000272  472444  006412  020040  020040  020040  020040  051525  041132  046040
000302  472454  020040  031454  031415  005040  020040  020040  020040  020101
000312  472464  047104  020040  020040  020063  026060  006412  020040  020040
000322  472474  020040  020040  046104  040440  020040  020040  031454  052056
000332  472504  055122  047415  005040  020040  020040  020040  020101  042104
000342  472514  020040  020040  020063  026060  006412  020040  020040  020040
000352  472524  020040  051524  040440  020040  020040  030454  052056  046523
000362  472534  043510  006412  020040  020040  020040  020040  045123  051040
000372  472544  020040  020040  040120  052524  041415  005040  020040  020040
000402  472554  020040  020114  042101  020040  020040  020062  026124  027115
000412  472564  032415  005015  005124  027115  051507  040472  020114  042101
000422  472574  020040  020040  020061  026124  027115  051507  044015  005040
000432  472604  020040  020040  020040  020115  047526  055114  020040  020061
000442  472614  026061  006412  020040  020040  020040  020040  046517  053114
000452  472624  020040  020040  030454  030415  005040  020040  020040  020040
000462  472634  020115  047526  046040  020040  020061  026061  006412  020040
000472  472644  020040  020040  020040  046517  053114  020040  020040  030454
000502  472654  030015  005040  020040  020040  020040  020114  042101  020040
000512  472664  020040  020063  026124  027115  051513  006412  020040  020040
000522  472674  020040  020040  040516  042040  020040  020040  031454  030015
000532  472704  005040  020040  020040  020040  020114  042101  020040  020040
000542  472714  020063  026124  027132  051117  006412  020040  020040  020040
000552  472724  020040  040504  042132  020040  020040  031454  030015  005040
000562  472734  020040  020040  020040  020123  052101  020040  020040  020061
000572  472744  026124  027115  051507  044015  005040  020040  020040  020040
000602  472754  020112  051522  020040  020040  020100  050125  052103  006412
000612  472764  020040  020040  020040  020040  044516  041440  020040  020040
```

51. UNLOCK,PASSWORD=<password>. Unlocks locked commands if the correct password is entered. Locked commands are initially locked when the DEBUGGER is entered via CONTROL-Q.

52. XREF,SYMBOL=<name>,IN PROC=<procedure name> - Displays cross-reference information for the specified symbol. This requires the availability of a SCHEMA file for the procedure. If no procedure is specified (default), the symbol is assumed to be global. If a procedure is specified, the symbol is assumed to be either local to that procedure or global.

VII. ABNORMAL TERMINATION OUTPUT

VII.1 FILE ERROR HANDLER

The routine FILERR is an error handler that decodes the File Entry Table (FET) information and formats it into a display. The program is called with the FET of the file that has gotten an error return from $CIO. It enables the programmer to see what went wrong from the FET information.

The routine displays the FET information and then prompts the user as follows:

'WOULD YOU LIKE A TRACEBACK? (Y OR N)'.

The default response is 'N' and the user is returned to the last active command file via a 'Restart Master'.

A response of 'Y' invokes the following prompt:

'PLEASE ENTER LEVEL # (1,2,3)='.

The meanings of the level numbers can be found in the write-up of TRACEBACK. After the user responds with a number, traceback is invoked as usual.

The display places the file information on the CRT in readable form. The file name, request code, and return code are displayed along with the file characteristics and status information. The last six words of the FET are displayed in octal format across, at the bottom of the display, so that the user can see the contents of the whole FET. These words are only important for limited applications; thus, they are not formatted.

The routine is invoked by the user program by a 'CALL FILERR(FNAME)', where FNAME is the name of the file that has caused a $CIO error return. FILERR must be declared as an external procedure [i.e., EXT PROC FILERR(FILE)].

VII.2 SYSTEM FAULT HANDLER

Programs hosted by EASY call the system fault handler when errors of a catastrophic nature are encountered. When a fault (other than breakpoint) occurs, a message is displayed on the CRT naming the line and procedure in which the fault occured, the fault number, and a short description of the fault.

The user is allowed several options at this point. An option is selected by typing the appropriate key. Be careful when typing the option because once the key is typed, the dump (or whatever is selected) starts immediately and there is no way to stop. The available options are:

Z - deadstart

0 - dump all ECB's

1 - 0 + traceback

2 - 1 + dump of locals

3 - 2 + dump of globals

4 - 3 + program space dump

Q - invoke debugger

The Q option is available only if the fault occurred at an interrupt level lower than the debugger. Once the debug module is placed in control, debug commands may be entered through the console keyboard. Section VI contains a complete description of these commands.

If there is not enough main store at the time of the fault to load the disk-resident fault handler, traceback, and the debugger, only the first message is displayed and EASY is re-deadstarted.

Appendix B describes the error codes.

VII.3 ECB DUMPS

Event Control Blocks contain information useful in determining abnormalities of a program. For a complete description of ECBs and how they are displayed, refer to Section VI, item 23, DISPLAYECB.

VII.4 TRACEBACK

Traceback provides a history of procedure calls and a dump of the run-time stack frame associated with each call. Traceback may also display the current status of global variables. Traceback use is explained in detail in Section VI, item 50.

REFERENCES

1. Nanodata Corporation, *QM-NCS, Preliminary Systems Operations Guide*, Williamsville, New York, May 1977.

2. Nanodata Corporation, *Task Control (TCP 1.05)*, Revision 2, Williamsville, New York, 1976.

3. John Perry, *SIMPL-Q Reference Manual*, Naval Surface Weapons Center, Dahlgren Laboratory Technical Report NSWC/DL TR-3778, Dahlgren, Virginia, April 1978.

4. Charles Flink, *Easy - The Design and Implementation of an Intermediate Language Machine*, Naval Surface Weapons Center, Dahlgren Laboratory Technical Report NSWC/DL TR-3765, Dahlgren, Virginia, December 1977.

5. Control Data Corporation, *CDC INTERCOM Reference Manual*, CDC 603071000, St. Paul, Minnesota, 1974.

6. John Perry, *Emulation Aid System (EASY) System Programmer's Guide*, Naval Surface Weapons Center, Dahlgren Laboratory Technical Report NSWC/DL TR-3774, Dahlgren, Virginia, December 1977.

7. K-74 Memos, Memo # 100, February 22, 1979.

APPENDIX A

HOW TO RUN EASY

HOW TO RUN EASY

The use of the EASY system requires a few comments about various controls on the QM-1 and its peripherals.

1. Power Mode Switch - The large switch on the silver console immediately in front of the CRT is the power switch. Three positions are important to the user.

   OFF - Power Off for Mainframe and Console
   RUN - Normal Operating Position
   IML - Initial Microload, or "Deadstart" position (see Deadstart below)

2. MC - The red pushbutton on the console close to the power switch is master clear. It is used only at deadstart (functions only if power switch is set to 'IML').

3. Start - The black pushbutton immediately below the master clear is the start button. Whenever the machine (QM-1) has reached a programmed stop or otherwise halted (e.g., after a master clear), the start button is pushed to continue.

4. PGM Stop - The program stop switch (and light indicating when it is on) is used to enable a QM-1 hardware stop. This switch should be off at all times.

5. Baud Rate - Behind the CRT is a rotary switch which selects the CRT speed. Only one setting is of importance to the user: 9600 baud. (At deadstart, 9600 baud is default. It should not be necessary for the user to change the baud rate.)

6. F Switches - On the QM-1 mainframe there is a board with two sets of three small, silver toggle switches arranged vertically. For the current EASY system, they should always be switched off (down). Do not change the settings of any other switches.

7. Disk Loader Switch - On the QM-1 mainframe there is a toggle switch which controls the disk loader. Up (the normal position) implies load from disk. Down implies load from tape cartridge.

8. Controls to Power Up Printer, Tapes, and Card Reader - The peripherals are easy to use, but the operation of them is best described in the context of the deadstart or power down sequence. If assistance is needed, any 6700 operator familiar with the MDS 2400 can help (the hardware is very similar).

9. Power Interruptions - In case of a power interruption while using the QM-1, notify the QM-1 Emulator Group of K74 (x 7854 - Hynson, Meyers, Naples, Hartung) for help if the computer will not run.

If no help is available, try the following procedures:

a. Check circuit breaker on inside door to mainframe and reset if tripped.

b. If it still does not run, power down CPU and then power up CPU.

c. If the disk fault lights on the small disk drive (9427) are on, power off, and then power on the drive. If the fault lights remain on, then it requires resetting two micro switches inside the disk drive. Do not attempt this unless you have been shown their location. The small disk drive is not used by EASY or DDT so these lights may be ignored if using these systems. This drive is used, however, by the TEMS system in Checkpoint - Restart.

The following gives the steps for bringing up the EASY system (i.e., deadstarting the QM-1, and the power down sequence to be followed when finished with the QM-1).

A. Deadstart Procedure

1. Power Up CPU - To bring up the QM-1, open the left front CPU cabinet door and turn on the power supply switch (labeled -5V PS) by depressing the red switch cover, so that the switch itself is in the down position. Next, flip the CPU circuit breaker (located just to the right of the power supply switch) to the 'ON' position. Select the 'RUN' position on the console control panel, and at this time the CPU should be powered up. If the display lights behind the right front CPU cabinet door are lit, the CPU is up. Else, depress the 'RESET' button at the lower left of the light display area, and if the CPU is still not up, call for assistance.

When the CPU has been left idle (Off) for a short period of time, it may be sufficient to power up the CPU by simply selecting the 'RUN' position on the console control panel. This is only effective when the power supply switch and main circuit breaker were not turned off when the CPU was last powered down.

2. Power Up CRT - The CRT display unit will normally be on when the CPU is powered up, and no intervention is required other than to ensure that the proper baud (9600) rate has been selected via the dial at the rear of the unit. If the CRT is on, the green 'POWER' light located on the keyboard will be lit. If the CRT is not on, turn the CRT power switch, located on the right side of the unit, to the 'ON' position. If the power light is still not lit, seek assistance. While using the display there are two (2) dials, located just beyond the CRT power switch, to control the contrast and brightness of the display.

3. Power Up Printer - The line printer can be powered up by turning the red switch to the 'ON' position, and can be brought on-line by depressing the 'START' button. To page eject, depress the 'STOP' button to make the printer off-line, then depress and hold down the 'FORM FEED' button until the page eject is accomplished. To achieve additional page ejects, release, depress, and then hold the 'FORM FEED' button. To line feed the printer, follow the same procedure as for a page eject: except depress, but do not hold down, the 'FORM FEED' button.

The line printer must be on, and on-line. If either of these conditions is not satisfied, a message will be posted on the CRT stating that the printer is not ready.

4. Power Up Tape Drives - To bring up the tape drives, it is necessary to open the rear cabinet door on each drive and depress the (black) 'POWER ON' button at the lower left corner of the rear of the tape unit. When depressed, the fan mechanism should come on to indicate that the drive has been powered up. Repeat the procedure for the other drive. The drives are labeled to indicate the associated logical unit numbers (0 or 1).

To use the tape drives, slide the glass door in the front of the unit down and mount the desired tape. Ensure that the tapes mounted are write-protected as desired (i.e., if to be read only, no write ring). The glass sliding door must be up (closed) for the drive to operate. Next, depress the 'ON' button at the top front of the drive and hold for 2 or 3 seconds. Press the 'LOAD POINT' selector and the tape should be positioned at load point (i.e., 'REMOTE' and 'LOAD POINT' lights should be on). If the tape should spin off the reel, remount the tape, and repeat the procedure. If the tape should continue to run forward, depress the 'REWIND' button, and after it rewinds, depress 'LOCAL' and then 'LOAD POINT' buttons. The tape should then be at load point. Unless otherwise specified, the normal operational mode of the drives is 800 BPI and can be selected by turning the dial to the 800 setting.

5. Spin Up Disk Drives - To spin up a new disk pack, first ensure that the power switch is off and that the drive motor is completely stopped. Lift the disk drive cover and place a disk casing over the old disk, turning counter-clockwise until a clicking noise occurs. Lift the disk up and secure it in its casing.

Next, release the bottom cover on the casing of the disk to be spun up. Hold the disk casing by the handle and place onto the drive motor assembly turning clockwise until it is fastened, then close the disk drive cover. The SELECT LOCK should be down and the READ ONLY switch should be up (points to WRITE). Then, turn on the 'STOP/START' switch: Wait until the 'ACCESS READY' light is lit.

6. Power Up the Card Reader - The card reader is powered up by depressing the 'POWER' button (left-most one). This button will be illuminated with a white light when there is power to the device and dark when the power is off. To enable the card reader for use, the 'RESET' button (right-most one) must be depressed. When activated, this button will be illuminated with a green light. To suspend a sequence of reading cards, depress the 'STOP' button which will be illuminated with a red light when active. To resume reading, depress the 'RESET' button.

When the system is trying to read a card and there are none in the hopper, the 'HOPPER CHECK' indicator and 'STOP' buttons will light up. To resume from this condition, place additional card input into the hopper and depress 'RESET'.

If the 'READ CHECK', 'PICK CHECK', or 'STACK CHECK' indicators light up, check for bad card punches or bent cards and start the last operation over again.

See Appendix I for card codes and description of End of File Card.

7. Bring up the EASY Emulator

a. If the disk loader switch is up, continue to step b. Otherwise, insert the EASY deadstart cartridge (current version) into the cartridge drive on the front of the control console.

b. Turn the console control panel selector to IML (Initial Micro Load)

c. Press MC and START buttons on the control panel.

d. If the disk loader switch is up, type EASY <CR>, else continue to step e.

e. An initial display will appear with instructions to enter date and time.

(1) Enter command DATE, respond with MM/DD/YY

(2) Enter command TIME, respond with HH:MM:SS

(3) Enter command @.

EASY is now ready.

B. Power Down Sequence

1. Power Down CPU - To power down the CPU for a short period of time (e.g., several hours or less), simply turn the left selector of the console control panel to 'OFF'. However, if the amount of time before bringing up the CPU is to be a day or more, or if unknown, then turn the selector off as above, open the left CPU cabinet door, and flip the CPU circuit breaker to the 'OFF' position. Finally, turn the power supply (labeled -5V PS) off (UP) and the CPU will be powered down.

2. Power Down CRT - The CRT display unit does not need to be turned off and simply loses power when the CPU is turned off.

3. Power Down Printer - Place the red switch in the off position.

4. Power Down Tape Drives - Dismount tapes as desired (to dismount from load point depress 'LOCAL', 'BACKWARD RUN', and the 'REWIND' buttons). Depress the 'OFF' selector button on the front top of the unit, and then open the rear cabinet door and depress the 'POWER OFF' button at the lower left corner of the tape drive cabinet. Repeat for the other drive.

5. Spin Down Disk Drives - Turn 'START/STOP' switch to stop.

6. Power Down Card Reader - Press POWER button.

APPENDIX B

ERROR MESSAGES

ERROR MESSAGES

For error numbers found in ECB 15 on an ECB Dump.

Category	Error #	Meaning
EASY	1	Unsatisfied external
EMULATOR	2	Statement/line executed in trace mode
	3	Call/return executed in trace mode
	4	Breakpoint encountered and enabled
	5	User executed halt instruction
	6	Attempted to move TS below BS
	7	Stack overflow
	10	Bad module
	11	Disk error loading program space
	12	Program space overflow
	13	Indirect link not referenced as halfword
	14	Illegal indirect link
	15	Address out of range
	16	Immediate found where address needed
	17	Operator found where operand needed
	20	Operand found where operator needed
	21	Array subscript out of bounds
	22	Illegal dope vector encountered
	23	Arithmetic overflow
	24	Field size error on part word instruction
	25	Starting bit error on part word instruction
	26	Privileged mode violation attempted
	27	Reserved code used with system instruction
	30	Invalid length input to substring instruction
	31	Invalid input to INTF instruction
	32	Invalid string comparison intrinsic
	33	Chain of entry points broken
	34	ECB referenced not legal for function
	35	Bad bit address of flag in SFLAG/CFLAG
	36	Bad parameter on string compare
	37	Dope vector for character array invalid
	40	Error on integer to string conversion
	41	Target of restart not parent proc
	42	Instruction not yet implemented
	43	Complete bit set on SIO
	44	Allocate error
	45	TCP function returned nonzero
	46	Unsupported TCP option specified in ECW
	47	Program check error
	50	Array too small

Category	Error #	Meaning
CIO	1001	Invalid CIO request
	1002	File already opened
	1003	No FNT slot available
	1004	File does not exist/file not opened
	1005	FNT/FET pointer mismatch
	1006	Exclusive read access error
	1007	EOD/Read or rewrite, EOE/Write
	1010	Disk hardware error
	1011	Write protect violation
	1012	Previous operation incomplete
	1013	Write after read
	1015	In, Out, Limit Error
	1016	Read after write
	1017	Invalid $FNTI Case Call
	1020	CIO cases that should not exist
	1022	Illegal expansion attempt
SEQUENTIAL I/O	1036	EOF during read
	1037	Invalid rewind request
	1040	Invalid end file request
	1041	EOF on write
PRINTER	1050	FATAL ERROR ON SIO CALL
	1051	INVALID LINE CONTROL PARAMETER
RANDOM I/O	1062	INVALID INDEX KEY
	1063	LENGTH TOO LONG ON REWRITE
	1064	INVALID REWRITE FLAG
	1065	NO DATA FOR INDEX KEY
	1066	INVALID NUMBER OF ELEMENTS PARAMETER
	1067	BAD STRING ARRAY ELEMENTS READ
WORD ADDRESSABLE	1074	INVALID WORD ADDRESS
	1075	INVALID NUMBER OF ELEMENTS PARAMETER
MISCELLANEOUS	1106	Limit exceeded on opened files
	1107	Invalid type of open
	1110	Illegal character found in $EXTST
	1111	Error found in $STEXT
CRT	1121	ERROR FOUND DURING CRT WRITE
CARD READER	1132	EOIC encountered during read
	1133	Invalid skip control
	1134	Card reader fatal error
SIMPL-Q ERROR	1144	SIMPL-Q ABORT - user called

APPENDIX C

SYSTEM COMMAND FILES

S/STEM COMMAND FILES

This is a listing of the System Command Files as of 6 November 1980. They are subject to change but are useful for reference and examples.

COMMANDS FOR EASY SYSTEM

```
200UT.  CDC 200UT (USERS TERMINAL) INTERACTIVE/RJE TERMINAL EMULATION.
LOAD THE 200UT EMULATOR CS AND MS, INITIALIZE AND START.
. (WARNING! USES 0 - 6000 (OCTAL) CS LOCATIONS AND 0 - 27000 (OCTAL) MS
UT200S. LOCATIONS.  MUST HAVE EASY CONFIGURED WITH IBS > 27000.)


. @>.
LEAVE THE CURRENT COMMAND FILE AND RETURN TO PARENT.

LEAVE.


BACKSPACE,UNIT=17,NUM OF RECORDS=999,PARITY=1.
BACKSPACES RECORDS ON A TAPE
0,1,1.
BKSP.


BIND.
INITIATES BIND, LINK EDITOR FOR EASY

BIND.


CALCULATE,EXPRESSION=SSSSSSSSSSSSSSSSSSS,BASE=99.
CALCULATE AN EXPRESSION, BASE IS DEFAULT INPUT AND OUTPUT BASE.
,8.  NO INPUT - REPEATEDLY PROMPTS FOR INPUT.
SCOMPU.


CIMPORT,TO FILE=SSSSSSSSSS,TRIM=!YES!NO!.
IMPORT CARD IMAGE FILE FROM THE CDC 6000
CDR,YES.
CIMPRT.


COMPARE,A=SSSSSSSSSS,B=SSSSSSSSSS,NUM FILES=999,PARITY=1,ERRS=!*LPT!*CRT!.
COMPARES FILE A WITH FILE B ,0-7,10-17 FOR TAPE, ERRORS GO ON ERRS
0,QM1,1,1,*LPT.
COMPAR.
```

```
COMPRESS,FROM=SSSSSSSSSSSS,TO=SSSSSSSSSSSS,NUM OF COLUMNS=99.
COMPRESSES MULTIPLE BLANKS AND DELETES 1-LINE COMMENTS,EJECT,SPACE,SKIP
QM1,SCR,72.
CMPRES.


COPY,FROM=SSSSSSSSSS,TO=SSSSSSSSSS.
COPIES FILES (CARDS OR DISK) TO DISK FILES
*CDR,CDR.  CARD READER TO TEMP DISK FILE
COPY.


COPYD,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP-ON=!EOD!EOE!.
FAST DISK  ---FILE---  COPY EITHER DATA OR FULL EXTENT
QM1,SCR,EOD.
$COPYD. ENTRY PROC FOR FAST DISK COPY


COPYNS,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=!END!ZEROS!ETX!EOE!.
COPY A NOVA FILE TO A SIMPLQ FILE
200UT:LPT,CDR,ETX.
COPYNS.


COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=!ZEROS!ETX!.
COPY A SIMPLQ FILE TO A NOVA FILE
CDR,200UT:CDR,ETX.
COPYSN.


CPX,CONFIGURATION FILE=SSSSSSSSSS.
RUN EXPERIMENTAL CONTROL PROGRAM (CPX) SUBSYSTEM.
.      NULL CONFIG FILE - DO NOT CONFIGURE
$CPX.


DATE,=19S79399.  (MM/DD/YY)
SET DATE.
01/00/00.
$SDATE.


DDT,SCHEMA #=!0!1!2!,UIC=SSS.
START DDT SUBSYSTEM.
0,   .
DDTENV.


DEADSTART.
RECONFIGURE AND DEADSTART EASY MACHINE
NO PARAMETERS - THIS COMMAND INVOKES A SUBSERVICE OF MASTER
$DDST$.
```

```
DIRECTORY,SEARCH 1ST=19,2ND=19,3RD=19,4TH DIR=19,REQ DIR=,REQ DIR=.
SELECT FILE DIRECTORY SEARCH ORDER (DRIVE=0-1, DIRECTORY=0-9)
,,,,03,00.          DRIVE/DIRECTORY DEFAULT VALUES
ULINK.   DRIVE ASSUMED 0 IF ONLY 1 DIGIT (0 OR 1) ENTERED


DISK-SAVE,PASSWD=SSSS,MTUNIT=17,MTFILE=99,DSKUNIT=7.
SAVE A DISK FILE ON TAPE
HELP,0,0,0.
DSKSV.


EDIT.
CRT BASED EDITOR  (COMMANDS EXPLAINED AS EDIT INITIAL DISPLAY)

EDIT.


EDITLIB,LIBRARY=SSSSSSSSSS,STATUS=!OLD!NEW!.
EDIT SYSTEM LIBRARIES
USERLIB,OLD.
EDTLIB.


EDITOR.
ENTER THE PARTITIONED DATA SET EDITOR.  (TTY ORIENTED)
.
$EDTOR.


EXEC,FILE=SSSSSSSSSS.
EXECUTE A FILE OF COMMANDS
*CDR.
EXEC.


FH-UTILITIES.
INVOKES THE FILE HANDLER'S UTILITY PACKAGE.
  NONE
FHDRVR,LOCKED.


FILES,FIRST USER=19.
ENTER FILES SUBSYSTEM -- MANIPULATE DIRECTORIES
04.
$FILES.


IMPORT,TO FILE=SSSSSSSSSS.
IMPORT BINARY FILE SHIPPED IN FROM CDC 6700.
QM1.
IMPORT.
```

```
LIBRARY,SEARCH 1ST=SSSSSSSSSS,2ND=SSSSSSSSSSS,3RD=SSSSSSSSSS.
SELECT LIBRARY SEARCH ORDER FOR ATTEMPTS TO SATISFY UNSATISFIED EXTERNALS.
SYSTEM,SYSTEM,SYSTEM.
$LIBRA.


LISTCF,COMMAND FILE=SSSSSSSSSS.
LIST COMMAND FILE ON PRINTER.
CMDTMP.
LISTCF.


LOCK.
LOCKS KEYBOARD.
  NONE
LOCKBD,LOCKED.


MACRO,OPTIONS=AAAAAAA,INPUT=SSSSSSSSSS,OUTPUT=SSSSSSSSSS.
PERFORM STANDALONE MACRO PASS
S,*CDR,CDR. LIST OPTION ON, USE N FOR NO LISTING
SML.


MAKECF,INPUT=SSSSSSSSSS,COMMAND FILE=SSSSSSSSSS.
MAKE A COMMAND FILE FROM COMMAND DECK.
*CDR,CMDTMP.     CARD READER TO TEMP FILE
MAKECF.


MAKERESIDENT,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS,=SSSSSS.
MAKES RESIDENT UP TO EIGHT MODULES REFERENCED BY ENTRY NAME.
,,,,,,,,.(=DBUGR,DBGSUP,MAP,STEPEZ,$TRACE,$FAULD. MAKES CONTROL-Q RESIDENT.)
$RSDNT.


MODULATE,BINARY=SSSSSSSSSS,FORMAT=!NCS!META!SMITE!,TO=SSSSSSSSSS,NAME=SSSSSS.
ENCAPSULATE AN EMULATOR FOR INGESTION BY EDITLIB. (GENERATE AN EMULATOR MODULE).
,NCS,QM1,EMXXXX.          (DEFAULTS CONVERT NCS BINARY INTO MODULE EMXXXX ON QM1.)
MDLATE.  (ONE BINARY INPUT FILE TREATED IF SPECIFIED,DEFAULT PROMPTS FOR MANY.)


MT-TO-DISK,UNIT=17,FILE #=99,TO FILE=SSSSSSSSSS,REWIND=!BOTH!BEFORE!AFTER!.
LOAD SIMPLQ BINARY FROM TAPE.
0,0,QM1,NONE.  DFAULT IS NO REWIND
MTTODS.


PASCAL,FLOPPY=SSSSSSSSSS,USER=99.
START PASCAL SYSTEM.
FLOPPY,08.
SETPAL.
```

```
PATCH-CALC.
CALCULATOR FOR EASY MACHINE CODE PATCHES

PACALC.


PHL,SENDER ID=7,EXEC FILE=SSSSSSSSSS.
INVOKE PATH HANDLER LANGUAGE SUBSYSTEM.
0,.     NULL EXEC FILE MEANS ACCEPT COMMANDS FROM KEYBOARD
PHL.


PRINT,FILE=SSSSSSSSSS,SCAN=!ON!OFF!.
LIST CARD IMAGES ON PRINTER, SCAN FOR /+EJECT+/ , ETC.
*CDR,OFF. (SCAN INCLUDES LINF NUMBERS AS WOULD BE LISTED BY THE COMPILER.)
PRSNT.


PRINT-SPOOL,FILE=SSSSSSSSSS.
PRINT SPOOL FILE FROM CPX.
200UT:CDR.
PRSP00.


PRU-MOD,FILE=SSSSSSSSSS,PRU #=777777,WORD #=777,VALUE=777777,COUNT=777.
MODIFY 'PRU' OF 'FILE' STARTING AT 'WORD' SET 'COUNT' WORDS = 'VALUE'.
QM1,0,0,0,0.   DEFAULTS CAUSE NO MOD, ALL #'S ARE OCTAL.
SMODPR,LOCKED.  IF TOO MANY WORDS ARE SPECIFIED 'COUNT' IS TRUNCATED.


PRUDMP,FILE=SSSSSSSSSS,SKIP=9999,DUMP=9999,TO=SAAA,TRANS=!E!6!8!,FMT=A99.
DUMP THE CONTENTS OF A DESIRED FILE.  (FMT OPTIONS=!018!036!D18!D36!X18!X36!)
QM1,0,9999,*LPT,6,018.     (TO OPTIONS=!*LPT!*CRT!BOTH!)
PRUDMP.


QCONTROL,MODE=!ON!OFF!.
ENABLE/DISABLE CONTROL-Q DEBUGGER.
OFF.  MUST BE EXPLICITLY TURNED ON
QCNTRL,LOCKED.


RDTAPE,UNIT=17.
READING TAPE FOR USE WITH 6000 AND PRINT.
0.
RDTAPE.


RESTORE-DISK,PASSWD=SSSS,MTUNIT=17,MTFILE=99,DSKUNIT=7.
RESTORE A FILE TO DISK FROM TAPE
0,0,0,0.
RESDSK.
```

```
REWIND,A=SSSSSSSSSS,B=SSSSSSSSSS,C=SSSSSSSSSS,D=SSSSSSSSSS,E=SSSSSSSSSS.
REWINDS FILES , ENTER 0-7,10-17 FOR TAPES.
0,,,,.
REWD.


SIMPLQ,OPTIONS=AAAAAAA,INPUT=SSSSSSSSSS,BINARY=SSSSSSSSSS,SCHEMA=SSSSSSSSSS.
PERFORM SIMPLQ COMPILATION
SFM1,*CDR,QM1,SCHEMA.
SIMPLQ.


SKIPBACKWARD,UNIT=17,NUM OF FILES=999,PARITY=1.
SKIPS BACKWARD FILES ON A TAPE
0,1,1.
SKIPB.


SKIPFORWARD,UNIT=17,NUM OF FILES=999.
SKIPS FORWARD FILES ON A TAPE
0,1.
SKIPF.


SM-TO-DISK,UNIT=17,FILE #=99,TO FILE=SSSSSSSSSS,REWIND=!BOTH!BEFORE!AFTER!.
COPY SMITE BINARY TO QM1 FILE.
0,0,QM1,NONE. DEFAULT IS NO REWIND
SMTODS.


SOURCE-TO-DISK,TAPE UNIT=17,DISK FILE=SSSSSSSSSS.
TRANSFER UPDATE SOURCE FILE TO QM1.
1,*DECK.  (*DECK OPTION USES FILENAME SPECIFIED ON *DECK CARDS INPUT)
SOUDSK.


SPY,EXPR=SSSSSSSSSSSSSSSSSSSS,MIN=777777,MAX=777777,BINSIZE=777777,MASK=777777.
SPY ON EXPRESSION DURING EXECUTION, PRINTS A HISTOGRAM OF VALUES ENCOUNTERED.
MS(0),0,777777,10000,777777.
SPYON.


SPY-PROC,# PROCS=9999.
START SPY ON PROCS AND MODULES
400,100.        100 MILLISECOND SPY INTERVAL
SPYPIN.


TERMSIM,LDID=77,BAUD=!300!600!1200!.
ENTER THE TERMINAL SIMULATOR SERVICE.
13,1200.
TERMSI.
```

```
TEST,ENTRY=ANNNNN,FILE=SSSSSSSSSS.
LOAD AND EXECUTE FILE (SIMPLQ PROGRAM).
TEST,QM1.
TEST$$.


TESTCF,COMMAND FILE=SSSSSSSSSS.
TEST A COMMAND FILE.
CMDTMP.
TESTCF.


TIME,=79S79S79.  (HH:MM:SS)
SET TIME.
23:60:60.
$STIME.


TRANS,UNIT=17,FILE=SSSSSSSSSS.
TRANSFER SOURCE FILES TO 9 TRACK TAPE FOR 6000.
0,CDR.
TRANSF.


UNLOAD,UNITA=17,UNITB=17.
UNLOADS TAPES
0,.
UNLD.


UNLOCK,PASSWORD=SSSSSSSSSS.
UNLOCKS KEYBOARD.
ESP.
UNLOCK.
```

COMMANDS FOR INITIAL TIME AND DATE SET.

```
<@>.
END OF INITIAL DISPLAY

INIFIN.

DATE,=19S79S99.  (MM/DD/YY)
SET DATE.
01/00/00.
$SDATE.

TIME,=79S79S79.  (HH:MM:SS)
SET TIME.
23:60:60.
$STIME.
```

```
<^E>.
     ESCAPE FROM 200UT SERVICE MODE.
.
UT2ESC.


<CR>.
     ESCAPE FROM 200UT SERVICE MODE.
.
UT2ESC.


<ESC>.
     ESCAPE FROM 200UT SERVICE MODE.
.
UT2ESC.


COPY,FROM=SSSSSSSSSS,TO=SSSSSSSSSS.
COPIES FILES (CARDS OR DISK) TO DISK FILES
*CDR,CDR.   CARD READER TO TEMP DISK FILE
COPY.


COPYNS,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=!END!ZEROS!ETX!EOE!.
COPY A NOVA FILE TO A SIMPLQ FILE
200UT:LPT,CDR,ETX.
COPYNS.


COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=!ZEROS!ETX!.
COPY A SIMPLQ FILE TO A NOVA FILE
CDR,200UT:CDR,ETX.
COPYSN.


DIRECTORY,SEARCH 1ST=19,2ND=19,3RD=19,REQ DIR=,REQ DIR=,REQ DIR=.
 SELECT FILE DIRECTORY SEARCH ORDER (DRIVE=0-1, DIRECTORY=0-9)
,,,03,04,00.          DRIVE/DIRECTORY DEFAULT VALUES
ULINK.    DRIVE ASSUMED 0 IF ONLY 1 DIGIT (0 OR 1) ENTERED


EXEC,FILE=SSSSSSSSSS.
EXECUTE A FILE OF COMMANDS
*CDR.
EXEC.
```

```
IMPORT,TO FILE=SSSSSSSSSS.
IMPORT BINARY FILE SHIPPED IN FROM CDC 6700.
QM1.
IMPORT.


KILL.
     KILL THE 200UT AND RETURN TO EASY.
     (THIS COMMAND DOES NOT ALLOW YOU TO WAIT FOR LOGOUT INFO.)
.
UT2KIL.


LOGIN,PROGRAMMER NAME=NNNNNNNNNN, ACCOUNT=NNNNNNNNNN.
     INITIATE QUICK LOGIN SEQUENCE.
N68PERRY,99KK33.
UT2LIN.


LOGOUT.
     LOGOUT ON THE 200UT, KILL IT, AND RETURN TO EASY.
     (THIS COMMAND ALLOWS THE USER TO WAIT FOR LOGOUT INFO FROM THE 6700.)
.
UT2LOU.
```

COMMANDS FOR BIND: LINK EDITOR FOR THE EASY SYSTEM

```
<@>.
TERMINATE BIND, PROVIDES A WARNING IF INVOKED WHEN THE CURRENT
  WORK FILE HAS NOT BEEN SAVED
BINDT.  SERVER IN BIND


DELETE,ENTRY=ANNNNN.
DELETES SPECIFIED ENTRY FROM ENTM CHAIN.
$$$$$$.     INVALID DEFAULT
BINDD.    SERVER IN BIND


INCLUDE,FILE=SSSSSSSSSS,# MODULES=9999.
INCORPORATE MODULES INTO WORK FILE.
QM1,9999.  DEFAULT IS ALL OF COMPILER OUTPUT
BINDI.  SERVER IN MASTER


MT-TO-DISK,UNIT=1,FILE #=9999,TO FILE=SSSSSSSSSS,REWIND=!A!B!.
LOAD SIMPLQ BINARY FROM TAPE
0,0,QM1, .     DEFAULT IS NO REWIND
MTTODS.
```

```
RESOLVE.
PERMANENTLY RESOLVES ALL SATISFIED EXTERNALS AND
  DELETES THE EXTM FROM THE CHAIN
BINDR.  SERVER IN MASTER


WRITE,FILE=SSSSSSSSSS.
COPIES WORK FILE TO SPECIFIED FILE.
QM2.    DEFAULT BIND OUTPUT
BINDW.  SERVER IN MASTER
```

COMMANDS FOR DEADSTART SERVICE (CMDDST)

```
<CR>.
PERFORM DEADSTART WITH LATEST DEADSTART TABLE

DDSTCR.


<ESC>.
ESCAPE DEADSTART, RETURN TO CALLER

LEAVE.


<SPACE>.
DISPLAY LATEST DEADSTART TABLE

DDSTSP.


DIRECTORIES,SEARCH 1ST=19,2ND=19,3RD=19,4TH=19,5TH=19,6TH=19.
RESETS CIO DIRECTORIES IN DEADSTART TABLE ONLY.
,,,,,,                          **** DOES NOT AFFECT CURRENT SETTINGS. ****
DDSTDI.


EASY-SPACE,BS=777777,TPS=777777.
RESET EASY MACHINE SPACE.
26,747777.
DDSTEZ.


LIBRARIES,SEARCH 1ST=SSSSSSSSSS,2ND=SSSSSSSSSS,3RD=SSSSSSSSSS.
RESETS LIBRARIES IN DEADSTART TABLE ONLY.
SYSTEM,SYSTEM,SIMPLQLIB.    **** DOES NOT AFFECT CURRENT SETTINGS. ****
DDSTLB.
```

```
SET-CSA,LENGTH=777777,WORKSPACE AT=777777.
SET LENGTH OF CONTROLSTORE A SEGMENT, PLACE EMULATOR WORKSPACE IN IT.
600,0.      VALID ONLY FOR CONFIGURATION OF A SON
DD$TCS.     INVALID FOR DEADSTART


STR-BREAKPOINT,STR=777777,BLINE=9999,BID=ANNNNN.
RESET STR AND BREAKPOINT REGISTERS.
0,*,$$$$$$.      DEFAULT BLINE DISABLES BREAKPOINTS.
DD$TBR.


                                EDITOR COMMAND FILE


<id>.
RETURN TO SCREEN ORIENTED EDIT MODE.
.
FINI.


<ESC>.
RETURN TO SCREEN ORIENTED EDIT MODE.
.
FINI.


CHANGE,OPTIONS=!V!G!VG!GV!.   (V=VETO,G=GLOBAL)
CHANGE OCCURRENCES OF ONE STRING TO ANOTHER.  COMMAND PROMPTS FOR FROM/TO
V.   DEFAULT IS VETO ON 1 MATCH.  SEARCH STARTS AT CURRENT CURSOR POSITI
EDCHG. GLOBAL IMPLIES ALL OCCURRENCES.  VETO GIVES USER CONTROL OVER CHAN


GETFILE,FILE=SSSSSSSSSS,FROM LINE=999999,TO LINE=999999.
INSERT FILE CONTENTS,GIVEN LINES SPECIFIED INTO EDIT FILE.
 ,1,999999. INSERTION BELOW LINE INDICATED BY CURSOR.
EDGET.    TERMINATES ON EOF ON FILE IF COUNT NOT EXPIRED.


HELP.
DISPLAY SCREEN ORIENTED EDITOR COMMANDS.
.
EDHELP.


QUIT,OUTPUT FILE=SSSSSSSSSS,TRIM=!Y!N!.
LEAVE EDITOR AND SAVE CHANGED TEXT ON FILE SPECIFIED, TRIMMED OR NOT.
 ,Y.  DEFAULT IS TO NOT SAVE CHANGED TEXT. TRIM=N CAUSES 80 COLUMN CARD
EDQUIT.       IMAGES TO BE OUTPUT.
```

```
RECOVERY.
RECOVER TEMPORARY EDITOR FILE FROM SCRATCH FILES.
.
EDRECV.


SET-MARGIN,AVERAGE=99,MINIMUM=99,MAXIMUM=99.
SET MARGIN PARAMETERS FOR JUSTIFY COMMAND.
60,1,80.
SET$MA.


TOP.
GO TO TOP OF FILE AND RETURN TO SCREEN ORIENTED EDIT MODE.
.
EDTOP.


WRITE,FILL=SSSSSSSSSS,LINES=9999,KILL=!Y!N!.
WRITE OUT LINES FROM EDIT BUFFER TO FILE, IF KILL THEN REMOVE LINES FROM
QM1,0,N.  IF THERE ARE NOT <LINES> LINES LEFT IN FILE, THE OUTPUT FILE IS
EDWRIT.  PADDED WITH BLANKS.
```

COMMANDS FOR EDITLIB. EDIT EASY SYSTEM LIBRARIES

```
<@>.
  LEAVE EDITLIB
  NO PARAMETERS
LEAVE.


ADD,FILE=SSSSSSSSSS,# MODULES=999.
ADD MODULES TO LIBRARY
QM1,1.
ELBADD.


BIND.
INITIATES BIND, LINK EDITOR FOR EASY

BIND.


BUILD,PASSWORD=SSSSSSSSSS.
BUILD OR REBUILD SYSTEM LIBRARY
FAILSAFE.  IS THE PASSWORD.  WARNING! THIS COMMAND DESTROYS THE OLD DIRECTORY.
ELBBUI.
```

```
COPY-MODULE,TO FILE=SSSSSSSSSS.
COPIES MODULES FROM LIBRARY TO THE SPECIFIED FILE.
QM1.      THE PROGRAM WILL PROMPT FOR MODULE NAMES.
ELBCOP.       ENTER A CARRIAGE RETURN TO FINISH.


DELETE,ENTRY=ANNNNN,TYPE=!P!D!.       P=PROC, D=DATA
DELETE ENTRY FROM DIRECTORY
X88888,P.      (DEFAULT IS DUMMY ENTRY GENERATED BY BIND... TYPICALLY DELETED.)
ELBDEL.


EXEC,FILE=SSSSSSSSSS.
EXECUTE A FILE OF COMMANDS
*CDR.
EXEC.


LIST,LEVEL=!1!2!3!4!5!.
LIST LIBRARY TABLES, STATISTICS, ETC.
2.  MODULE NAMES AND ENTRY NAMES
ELBLIS.


MT-TO-DISK,UNIT=1,FILE #=9999,TO FILE=SSSSSSSSSS,REWIND=!A!B!.
LOAD SIMPLQ BINARY FROM CDC TAPE
0,0,QM1, .     DEFAULT IS NO REWIND
MTTODS.


REMOVE,MODULE=ANNNNN.
REMOVE A MODULE FROM THE LIBRARY
$$$$$$.  DON'T USE THE DEFAULT
ELBREM.


REPLACE,FILE=SSSSSSSSSS,# MODULES=999.
REPLACE MODULES IN LIBRARY, ADD IF NOT IN LIBRARY
QM1,1.
ELBREP.


SMASH.
COMPACT THE LIBRARY
   NO PARAMETERS
ELBSMA.
```

COMMANDS FOR BUILD. BUILDING SYSTEM LIBRARIES

```
<@>.
LEAVE BUILD
  NO PARAMETERS
LEAVE.


INCLUDE,FILE=SSSSSSSSSS,# MODULES=999.
INCLUDE MODULES INTO LIBRARY DURING BUILD
QM1,1.
ELBINC.


MT-TO-DISK,UNIT=1,FILE #=9999,TO FILE=SSSSSSSSSS,REWIND=!A!B!.
LOAD SIMPLQ BINARY FROM CDC TAPE
0,0,QM1, .    DEFAULT IS NO REWIND
MTTODS.
```

CRT EDITOR (EDIT) COMMAND FILE.

```
<@>.
RETURN TO EASY.

FINI.


CREATE-PDS,FILE=SSSSSS.
CREATES A PDS IN FILE
SOURCE.
SEDCFI,LOCKED.


DELETE-PDS,FILE=SSSSSS.
REMOVES PDS ENTRIES FROM FILE.
SOURCE.
SEDDEL.


EXISTENT.
USED TO EDIT AN EXISTING EDIT FILE

SEDEXT.


GARBAGE,FILE=SSSSSS.
GARBAGE COLLECTS PDS FILES
SOURCE.
SEDGAR.
```

```
NEW.
BUILDS A NEW FILE STARTING WITH AN EMPTY BUFFER

SEDNEW.


OLD,FILE=SSSSSS,PDS=SSSSSS.
USED TO ACCESS A FILE TO BE EDITED
SOURCE,     .
SEDOLD.


SAVE,FILE=SSSSSS,PDS=SSSSSS.
SAVES EDITED VERSION.
SOURCE,     .
SEDSAV.


                    COMMANDS FOR FILE HANDLER'S UTILITIES.


<id>.  TERMINATE FILE HANDLER UTILITIES.
GET OUT OF THIS CMD FILE.

$SQUIT.


CREATE,ENTRY-NAME=SSSSSSSSSS. TO CHECKOUT FHCRE8E ROUTINE, ASSUMES STD.BUFFER...
ASSUMES THAT QM1 HAS BEEN INITIALIZED APPROPRIATELY FOR CHECKOUT.
ENTRYN1. THIS ROUTINE FOR CHECKOUT ONLY.
$SCRE8. MAY BE REMOVED FROM LIBRARY WHEN THRU IF DESIRED.


DEL-ENTRY,ENT-NAME=SSSSSSSSSS. TO CHECKOUT FHDELE ROUTINE..
ASSUMES THAT QM1 HAS BBEN INITIALIZED APPROPRIATELY FOR CHECKOUT.
ENTRYN1.  THIS ROUTINE FOR CHECKOUT ONLY.
$SDELE.


FIND-ENTRY,ENT-NAME=SSSSSSSSSS. TO CHECKOUT FHFINDE ROUTINE.
ASSUMES THAT QM1 HAS BEEN INITIALIZED APPROPRIATELY FOR CHECKOUT.
ENTRYN1.  THIS ROUTINE FOR CHECKOUT ONLY.
$SFINE.


G-COLLECT,FILE-NAME=SSSSSSSSSS.
INVOKES FILE HANDLER'S GARBAGE COLLECTION FACILITY.
QM1. DEFAULTS.
FHGCOL.


INITIALIZE,FILE-NAME=SSSSSSSSSS.
INVOKES FILE HANDLER'S INITIALIZE FACILITY.
QM1.
FHINIT.
```

```
LIST-DIR.   TO CHECKOUT FHLIST ROUTINE..
ASSUMES THAT QM1 HAS BEEN INITIALIZED APPROPRIATELY FOR CHECKOUT.

$$LSTD.


LST-ENT-INFO,INDEX=9999.     TO CHECKOUT LSTENTRY ROUTINE..
ASSUMES THAT QM1 HAS BEEN INITIALIZED APPROPRIATELY FOR CHECKOUT.
2.   THIS ROUTINE STRICTLY FOR CHECKOUT.
$$LSTF.


OPEN.  CHECK OUT FILE HANDLER'S OPEN ROUTINE.
ONLY FOR CHECKOUT PURPOSES.

$$OPEN.


RET-STR,INDEX=9999,REC#=9999.
TO CHECKOUT FHNTHSTR ROUTINE.
1,1.
$$NSTR.


UPDATE,INDEX=9999,OPR=A,OLD=SSSSSSSSSSSSSSSSSS,NEW=SSSSSSSSSSSSSSSSSS.
TO CHECKOUT FHUPDTE ROUTINE, ASSUMES STD. BUFFER & FILE QM1 INITIALIZED APPORP.
1,D,OLD-TEST-STRING,NEW-TEST STRING. DEFAULTS, INT, STRING, STRING, STRING.
$$UPDT. CAN BE REMOVED FROM LIBRARY WHEN THRU IF DESIRED.
```

COMMANDS FOR FILES DIRECTORY MANIPULATION (CMDFILES)

```
E*.
TERMINATE FILES PROCESSING.
.
LEAVE.


ADD,FILE NAME=SSSSSSSSSS,LENGTH=77777.
ADD NEW FILE TO DIRECTORY, LENGTH SPECIFIED IN NOVA SECTORS
$,300.
FSADD.


CHANGE,FILE=SSSSSSSSSS,NEW NAME=SSSSSSSSSS,EODPRU=77777,EODWORD=177,PERM=!W!R!.
CHANGE FILE ATTRIBUTES, DEFAULT OF * LEAVES ATTRIBUTE UNCHANGED.
$,*,*,0,*.
FSCHNG.
```

```
COMPACT,PASSWORD=SSSSSSSSSS.
COMPACTS CURRENT USER (ELIMINATES HOLES BETWEEN FILES).
WRONG.      DANGEROUS OPERATION -- DISKSAVE SHOULD BE PERFORMED FIRST.
F$COMP,LOCKED.


DELETE,FILE NAME=SSSSSSSSSS.
DELETE SPECIFIED FILE, USER IS PROMPTED FOR VERIFICATION BEFORE DELETION.
$.
F$DEL.


DISPLAY,FILE NAME=SSSSSSSSSS.
DISPLAY DIRECTORY ENTRY FOR A PARTICULAR FILE.
$.
F$DISP.


INITIALIZE,USER=19,PASSWORD=SSSSSSSSSS.
CLEARS DIRECTORY FOR NEW USER, CREATES FILE ENTRY FOR DIRECTORY.
XX,WRONG.   USER WILL BE PROMPTED FOR VERIFICATION TWICE.
F$INIT,LOCKED.


LOCK.
LOCK CONSOLE AFTER COMPLETION OF PROTECTED COMMANDS.
.
LOCKBD,LOCKED.


MOVE,FILE=SSSSSSSSSS,START CYL=777,SECTOR=77,LENGTH=77777,MOVE DATA=!YES!NO!.
MOVE FILE TO NEW LOCATION AND/OR CHANGE EXTENT, POSSIBLY MOVING FILE CONTENTS.
$,*,0,*,NO.     DEFAULT OF * LEAVES CORRESPONDING ATTRIBUTE UNCHANGED.
F$MOVE,LOCKED.


OVERLAPS.
DISPLAY OVERLAPS
.
F$OVER.


REPORT,TO=!CRT!PRINTER!BOTH!,SORTED BY=!NAME!POSITION!,SUBFILES=!YES!NO!.
DISPLAY CURRENT DIRECTORY.  SUBFILES ARE FILES NESTED WITHIN ANOTHER FILE.
CRT,POSITION,YES.
F$REPO.


UNLOCK,PASSWORD=SSSSSSSSSS.
UNLOCK CONSOLE FOR PROTECTED COMMANDS.
WRONG.
UNLOCK.
```

```
USER,CHANGE TO=19.
CHANGE USER (WORKING DIRECTORY).  DEFAULT DISPLAYS CURRENT USER
*.
FSUSER.

                    COMMANDS FOR EASY MACHINE CODE PATCH CALCULATOR

<@>.


LEAVE.


CODE-NAME,NAME=ANNNNN,TYPE=!DATA!PROC!.
ENCODE ENTRY NAME INTO 36-BIT INTEGER
$$$$$$,PROC.
PCCODN.


DECODE-NAME,VALUE=777777777777.
DECODE 36-BIT INTEGER INTO SYMBOLIC NAME
333333333333.
PCDECN.


IMMEDIATE,VALUE=FFFFFF,BASE=!B!O!D!X!.
CALCULATE IMMEDIATE OPERAND, VALUE GIVEN IN SPECIFIED BASE.
0,0.
PCIMED.


INDIRECT-LINK,AT=777777,TO POINT TO=777777.
CALCULATE INDIRECT LINK FROM ONE ADDRESS TO ANOTHER.
0,0.
PCINDL.


PC-REL-ADDRESS,AT=777777,ADDRESS=777777,INDIRECT=!YES!NO!,HALFWORD=!YES!NO!.
CALCULATE A PC-RELATIVE ADDRESS
0,0,NO,NO.
PCPCRL.


PRU-ADDRESS,ADDRESS=777777.       GIVEN 18-BIT WORD ADDRESS INTO
  A FILE, WILL RETURN THE PRU NUMBER (DECIMAL AND OCTAL)
200.       AND THE WORD ADDRESS INTO THE PRU.
PCPRUA.
```

```
STACK-ADDRESS,ADDRESS=777777,INDIRECT=!YES!NO!,HALFWORD=!YES!NO!.
CALCULATE A STACK ADDRESS
0,NO,NO.
PCSTAK.
```

PATH HANDLER TEST COMMANDS (CMDPHTEST)

```
<@>.
END OF PATH HANDLER RUN.

PHTOFF.


CHECK,BUF=9,FOR PATTERN=9,# WORDS=77.
COMPARE BUFFER CONTENTS AGAINST STAND ..D PATTERN.
0,0,100.
PHTCHE.


COPY,TO BUF=9,PATTERN=9,# WORDS=77.
COPY STANDARD PATTERN INTO BUFFER.
0,0,100.
PHTCOP.


DUMP,BUF=9,WORD COUNT=77.
DUMP BUFFER CONTENTS TO PRINTER.
0,100.
PHTDUM.


HIO,PATH/SUBCH=777.
HALT I/O ON GIVEN PATH AND SUBCHANNEL.
000.
NOTI.  TCP OVERLAY FOR HIO NOT YET DELIVERED


OPTIONS,DISPLAY=!CRT!PRINTER!BOTH!.
SET DISPLAY DESTINATION.
BOTH.
PHTOPT.


SET-DATA,BUF=9,WORD=77,CONTENTS=777777.
SET A DATA WORD IN A BUFFER.
0,0,0.
PHTSET.
```

```
SIO-NTDS,ECB=9,COW=77,PATH/SUBCH=777,BUF=9,WCR=77,C=1,CW=99,XW=99.
START NTDS I/O ON GIVEN ECB WITH SPECIFIED TIOCB INFORMATION.
1,03,000,0,100,0,18,18.
PHTSIO.


SIO-UYK7,ECB=9,COW=77,PATH/SUBCH=777,DO=777777,D1=777777,D2=777777,D3=777777.
START UYK-7 I/O ON GIVEN ECB WITH SPECIFIED COMMAND AND DATA.
1,13,000,000000,000000,000000,000000.
PHTSI7.


TIOCB,ECB=9.
DISPLAY ECB IN TIOCB FORMAT.
1.
PHTIOC.


WAIT,ECB=9,FOR TROUBLE=77.
WAIT FOR ECB POSTED AND CHECK FOR PROPER TROUBLE RESPONSE.
1,000.
PHTWAI.
```

COMMAND FILE FOR TERMINAL SIMULATOR

```
<@>.
TERMINATE TERMINAL SIMULATOR.
.
QUIT.


<CR>.
ENTER TERMINAL SIMULATOR MODE.
.
SIMODE.


COPYNS,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,STOP ON=!END!ZEROS!ETX!EOE!.
COPY A NOVA FILE TO A SIMPLQ FILE.
QM1,CDR,ZEROS.
COPYNS.


COPYSN,FROM=SSSSSSSSSS,TO=SSSSSSSSSS,END WITH=!ZEROS!ETX!.
COPY A SIMPLQ FILE TO A NOVA FILE.
CDR,QM1,ZEROS.
COPYSN.
```

```
ECHO,=!ON!OFF!.
SELECT LOCAL ECHO OPTION.
OFF.
SETECH.


EDIT.
CRT BASED EDITOR.

EDIT.


ESCAPE-RESET.
SET ESCAPE CHARACTER.
.
TRMSET.


FROM-FILE,=SSSSSSSSSS.
SET UP DESIRED FROM FILE FOR TERMINAL SIMULATOR.
OFF.
SETFRO.


QUIT.
TERMINATE TERMINAL SIMULATOR.
.
QUIT.


SIM-MODE.
ENTER TERMINAL SIMULATOR MODE.
.
SIMODE.


STATUS.
TERMINAL SIMULATOR STATUS DISPLAY.
.
TERMST.


TO-FILE,=SSSSSSSSSS.
SET UP DESIRED TO FILE FOR TERMINAL SIMULATOR.
OFF.
SETTO.
```

SAVE/RESTORE COMMAND FILE (CMDSVRS)

```
<@>.
  LEAVE SAVE-RESTORE
  NO PARAMETERS
FINSVR.


CALC-NOVA,START CYL=777,START SEC=77,FINISH CYL=777,FINISH SEC=77.
CALCULATE THE SIZE IN OCTAL NOVA SECTORS OF THE SPACE FROM "START" THRU
0,0,617,27. "FINISH". DEFAULTS = <1 FULL USER & SHOW MINIMUM & MAXIMUM
$NOVAC,LOCKED.       TYPE-IN VALUES.


NOVA-FORMAT,USER=9,CYL=777,SEC=77,SIZE=777777.
ADDRESS IN NOVA FORMAT
0,0,0,0.
NOVADD.


PHYS-FORMAT,CYL=999,HS=99,SEC=99,SIZE=99999.
ADDRESS IN PHYSICAL FORMAT
0,0,0,0.
PHYSAD.


USER-FORMAT,USER=9.
USER ADDRESS
0.
USERAD.
```

CPX COMANDS

```
<@>.

.
CPX$TT.


<SPACE>.
SAME AS STATE DISPLAY   (PROCESSORS)

CPX$SD.


ASSIGN,DEVICE=!CON2!CON3!RDR!PRTR!TAPE!CLCK!LINE!DISK!,TO PROCESS=7,AS LDID=77.
ASSIGN A REAL DEVICE TO A PROCESSOR
DISK,1,10.
CPX$AS.
```

```
BUFFER-MAP,PROC=7,ARRAY=SSSSSS,SIZE=777777.
INCREASE BUFFER SIZE OF PERIPHERAL SIMULATIONS.
1,DUMMY,2.
BMAP.


CHECKPOINT,TO=SSSSSSSSSS,MESSAGE=SSSSSSSSSSSSSSSSSS.
CHECKPOINT THE STATE OF EXPERIMENT.
CHECKPOINT,***CHECKPOINT***.
$CHKPT.


CONNECT,PATH=77,TO PROCESSOR=7,AS LDID=77.
CONNECT A PATH TO A PROCESSOR.
00,1,00.
CPX$CN.


CREATE-MEMORY,NAME=SSSSSSSSSS,WORD SIZE=99,LENGTH=777777.
CREATE A TARGET MACHINE MEMORY.
MEMXXX,18,200000.  DUMMY DEFAULTS
CPX$CM.


CREATE-PATH,PATH=77,PATH TYPE=!UYK-7!PERIPHERAL!INTERCOMPUTER!.
CREATE A PATH OF THE GIVEN TYPE.
00,PERIPHERAL.
CPX$CQ.


CREATE-PROCESS,PROCESSOR=/,MEMORY=SSSSSSSSSS,MACHINE TYPE=ANNN.
CREATE A PROCESSOR, CONNECT IT TO AN EXISTING MEMORY.
1,MEMXXX,EASY.
CPX$CP.


DELETE-MEMORY,NAME=SSSSSSSSSS.
DELETE A TARGET MACHINE MEMORY.
MEMXXX.
CPX$DM.


DELETE-PATH,PATH=77.
DELETE A PATH, BREAK ALL DISCONNECTIONS.
00.
CPX$DQ.


DELETE-PROCESS,PROCESSORS=7777777.
DELETE A PROCESSOR(S), RELEASE ITS DEVICES
*.       USE PREVIOUS
CPX$DP.
```

```
DEVICE-MAP,PROC=7,ARRAY=SSSSSS,VTOR=7777777777.
MAP VIRTUAL TO REAL DEVICES.
1,DUMMY,0.
DMAP.


DISCONNECT,PATH=77,FROM PROCESSOR=7,FROM LDID=77.
DISCONNECT A PATH FROM A PROCESSOR.
00,1,.   NULL LDID --> BREAK ALL CONNECTIONS OF THIS PATH TO THIS PROCESSOR.
CPX$DC.


DISPLAY-DEVICE.
DISPLAY ALLOCATION OF REAL DEVICES.

CPX$DD.


DISPLAY-MEMORY.
DISPLAY TARGET MACHINE MEMORIES.
.
CPX$MD.


DISPLAY-OLDPT.
OLD PATH TABLE DISPLAY.

CPX$PD.


DISPLAY-PATHS,PATH NUMBER=77.
CPX PATH DISPLAY. DEFAULT GIVES GENERAL DISPLAY, OPTION GIVES DETAILED
ALL.DISPLAY ON ONE PATH. OLD PATH DISPLAY IS AVAILABLE AS
CPX$PT.  DISPLAY-OLDPT.


DISPLAY-STATE.
DISPLAY STATE OF ALL PROCESSORS.

CPX$SD.


EFIOC7,PICK-IT=7.
TEMPORARY KLUDGE**********
2.
IOC7XF.


EFIUYK7,PICK-IT=7.
TEMPORARY KLUDGE**********
1.
IUYK7XF.
```

```
EXEC,FILE=SSSSSSSSSS.
EXECUTE A FILE OF COMMANDS
CDR.
EXEC.


FLOAT-CONSOLE,TO PROCESSOR=7,CONTROL CHARS GO TO=!SON!PARENT!.
LOAN CONSOLE 1 TO A PROCESSOR WITH A VIRTUAL CONSOLE.
1,SON.
CPX$FC.


FREEZE,PROCESSORS=7777777.
STOP A PROCESSOR(S), PRESERVE ITS PRESENT STATE  (SET FREEZE FLAG)
*.     USE PREVIOUS
CPX$FR.


GO,PROCESSORS=7777777.
START (OR RESTART) PROCESSORS.  (PLACE INTO GO STATE, CLEAR STEP FLAG)
*.          USE PREVIOUS PROCESSOR STRING
CPX$GO.


LDID-MAP,OF PROCESSOR=7.
DISPLAY LDID MAPPING FOR A PROCESSOR.
1.
CPX$LD.


MASTER, COMMAND FILE=SSSSSSSSSS.
INVOKE MASTER WITH COMMAND FILE SPECIFIED (ENTER SUBSYSTEM UNDER CPX.)
CMDSYS. (DEFAULT IS EASY SYSTEM COMMAND FILE. CAUTION! USE NO TAPES!)
TESTCF. (CPX REUSES THE TAPE ECB THEREFORE AVOID SUBSYSTEM TAPE OPERATIONS.)


MONITOR-DUMP,FILE=SSSSSSSSSS.
DUMP STATISTICS TO DISK WITHOUT STOPPING MONITOR.
.
DUMP$S.


MONITOR-OFF,FILE=SSSSSSSSSS.
TURN OFF PATH I/O MONITOR. DUMP STATISTICS TO SPECIFIED FILE
.
OFF$MO.


MONITOR-ON,DEBUG PRINT=A.
ACTIVATE PATH I/O MONITOR. USE OF THIS COMMAND WITH MONITOR ALREADY
NO.  ON SET DEBUG PRINT AS SPECIFIED
ON$MON.
```

```
MONITOR-PRINT,FILE=SSSSSSSSSS.
PRINT STATISTICS. IF FILE = NULL THEN PRINT CURRENT STATISTICS
.
STAT$P.


MONITOR-RESTART,FILE=SSSSSSSSSS,DEBUG PRINT=A.
CHECKPOINT RESTART MONITOR FROM SPECIFIED FILE. USE OF COMMAND WITH
,N. MONITOR ON SETS DEBUG PRINT AS SPECIFIED.
RES$MO.


PRINT-SPOOL,FILE=SSSSSSSSSS.
PRINT SPOOL FILE FROM CPX.
200UT:CDR.
PR$P00.


RELEASE,DEVICE=!CON2!CON3!RDR!PRTR!TAPE!CLCK!LINE!DISK!,FROM PROCESSOR-7
RELEASE A DEVICE
DISK,1.  PROCESSOR NUMBER USED ONLY WHEN RELEASING DISK.
CPX$RL.


RESTART,FROM=SSSSSSSSSS.
RESTART A PREVIOUS EXPERIMENT.
CHECKPOINT.
$RSTRT.


STEP,PROCESSORS=7777777.
STEP PROCESSORS (PLACE INTO GO STATE, SET STEP FLAG)
*.
CPX$ST.


STORAGE-MAP.
MAP STORAGE (MAINSTORE AND CONTROLSTORE)

CPX$SM.


SYNCHRONIZE,PROCESSORS=7777777,INTERVAL=999999.  (1 TICK = 100 MICROSEC)
SYNCHRONIZE PROCESSORS ON SPECIFIED INTERVAL (NUMBER OF CLOCK TICKS)
,10000.      DEFAULT = NULL  -->  TURNS SYNC OFF
CPX$SY.


THAW,PROCESSORS=7777777.
RELEASE PROCESSOR(S) FROM FREEZE   (CLEAR FREEZE FLAG)
*.     USE PREVIOUS
CPX$TH.
```

```
UYK20CPEF,SON=7.
START UYK-20 CPU
1.
YK20XF.


UYK20IOEF,SON=7.
START UYK-20 IOC
2.
IO20XF.


VIRTUAL-ASSIGN,DEVICE=!SYNC!CON!PRTR!RDR!,TO PROC=7,AS LDID=77,SPOOL=SSSSSSSSSS.
CREATE A VIRTUAL DEVICE FOR THE PROCESSOR.
SYNC,1,77,.
CPX$VA.


VIRTUAL-RELEAS,DEVICE=!SYNC!CON!PRTR!RDR!,FROM PROCESSOR=7.
DELETE A VIRTUAL DEVICE.
SYNC,1.
CPX$VR.
```

EASY DEBUGGING SERVICES

```
<*>.
DISPLAY SHORT PROGRAM SPACE MAP (SUBSET OF MAPPROGSPACE.)
1,NO,10.
MAP.


<+>.
PAGE DISPLAY FORWARD

PLUS.


<->.
PAGE DISPLAY BACKWARD

MINUS.


<@>.
END OF DEBUGGING

LEAVE.
```

```
<CR>.
REISSUE LAST MEMORY DISPLAY

ECHO.


<LF>.
PRINTS CURRENT DISPLAY

PRINT.


<PERIOD>.
DISPLAY SHORT MAP OF STACK  (SUBSET OF MAPSTACK.)
NO,15.
STKMAP.


ARRAY,ADDRESS=SSSSSSSSSS,FIRST ELEMENT=9999999.
DISPLAY SIMPL-Q ARRAY
0,0.
ARRDIS.


BASE,ADDRESS/MODULE=SSSSSSSSSS.
SET BASE ADDRESS FOR MAIN STORE DISPLAYS
0.  USE ACTUAL VALUE, /MODULE FOR MODULE, +PROC FOR STACK FRAME
BASET.


C,ADR=NSSSSSSSS,FORMAT=!O!OF!D!DF!X!XF!.
DISPLAY CONTROL STORE STARTING AT ADR.   (O=>OCTAL,D=>DECIMAL,X=>HEX)
0,0.       F FOR FULLWORD (36-BIT) FORMAT.
CSDIS.     IDENTICAL TO DISPLAYCS.


CALCULATE,EXPRESSION=SSSSSSSSSSSSSSSSSSS,BASE=99.
CALCULATE AN EXPRESSION, BASE IS DEFAULT INPUT AND OUTPUT BASE.
,8.  NO INPUT - REPEATEDLY PROMPTS FOR INPUT.
$COMPU.


CHANGE,SYMBOL=ANNNNNNNNNNNNNNNNNNN,IN PROC=ANNNNNNNNNNNNNNNNNNN.
ACCESS SCHEMA FOR TYPE AND LOCATION, PROGRAM WILL PROMPT FOR THE NEW VALUE.
JOE,.
CHANGE.


CHANGECS,ADDRESS=NSSSSSSSS,VALUE=NSSSSSSSSSSSSSSS,REP=999.
PLACE DATA IN CONTROL STORE MEMORY
0,0,1.
CSTORE,LOCKED.
```

```
CHANGECSM,ADR=NSSSSSSSS,VALS=NSSSSSSSSSSSSSSSSS.
STORE MULTIPLE CONTROL STORE LOCATIONS
0,0.    VALUES SEPARATED BY A /
MCSTOR,LOCKED.


CHANGEMS,ADDRESS=NSSSSSSSS,VALUE=NSSSSSSSSSSSSSS,REP=999999.
PLACE DATA IN MAIN STORE MEMORY
0,0,1.
MSTORE,LOCKED.


CHANGEMSM,ADR=NSSSSSSSS,VALS=NSSSSSSSSSSSSSSSSS.
STORE MULTIPLE MAINSTORE LOCATIONS
0,0.   VALUES SEPARATED BY A /
MMSTOR,LOCKED.


CHARACTER,ADDRESS=SSSSSSSS.
DISPLAY SIMPLQ CHARACTER VARIABLE
0.
CHADIS.


CODE,MODULE=SSSSSS,LINE=NNNN.
DISPLAY EASY MACHINE CODE IN SYMBOLIC FORMAT.
DDSTRT,1.
NOTI,LOCKED. (NOT YET IMPLEMENTED)


CODEPATCH,MODULE=SSSSSS,ADDRESS=NSSSSSSSS,CODE=SSSSSSSSSSSSSSS.
PATCH A MODULE VIA SYMBOLIC MACHINE CODE.
*,0,HALT.
NOTI,LOCKED. (NOT YET IMPLEMENTED)


CONVERT,THIS=NSSSSSSSSSSSSS,TO BASE=!D!B!O!X!.
CONVERT FROM ONE BASE TO ANOTHER (DECIMAL,BINARY,OCTAL,HEX)
X'FF',D.              CONVERSION TO DECIMAL IS DEFAULT
CONVRT.


DISPLAY,SYMBOL=ANNNNNNNNNNNNNNNNNN,IN PROC=ANNNNNNNNNNNNNNNNNN.
ACCESS SCHEMA FOR TYPE AND LOCATION THEN DISPLAY SYMBOL.
JOE,.
DSPLAY.


DISPLAYCS,ADDRESS=NSSSSSSS,FORMAT=!O!OF!D!DF!X!XF!.
DISPLAY CONTROL STORE MEMORY
0,0.
CSDIS.
```

```
DISPLAYECB,ECB NUM=17,PRINT=!YES!NO!.
DISPLAY EASY MACHINE ECBS
,NO.
ESPECB.


DISPLAYMS,ADDRESS=NSSSSSSSS,FORMAT=!O!OF!D!DF!X!XF!.
DISPLAY MAIN STORE MEMORY STARTING AT ADDRESS.  (O=>OCTAL,D=>DECIMAL,X=>HEX)
O,O.        F FOR FULLWORD (36-BIT) FORMAT.
MSDIS.


EXEC,FILE=SSSSSSSSS.
EXECUTE A FILE OF COMMANDS.
CDR.
EXEC.


GO.
CLEAR STEP OR HALT SETTINGS. (EXECUTION RESUMES ON LEAVING CONTROL-Q.)
*.          (SUBSET OF STEP FUNCTION.)
STEPEZ.


GOTO,LINE=NNNN.
CHANGE NEXT-STATEMENT-TO-EXECUTE.
*,O.    *=>LAST IDENTIFIED MODULE
NOTE,LOCKED. (NOT YET IMPLEMENTED)


BRKE,AT LINE=9999,IN PROC=NNNNNN.
SET EASY BREAKPOINT LOCATION
,SSSSSS. DEFAULTS.
BRKE.


INTEGER,ADDRESS=SSSSSSSS.
DISPLAY SIMPLQ INTEGER VARIABLE
.
INTEIS.


LINE,NUMBER=NNNN,FOR N=NN.
DISPLAY N LINES OF SOURCE STARTING AT LINE SPECIFIED.
,I. (SOURCE FILE SPECIFIED BY SCHEMA OR SOURCE COMMAND.
NOTE. (NOT YET IMPLEMENTED)


LOCK.
RELOCKS THE KEYBOARD

LOCKED,LOCKED.
```

```
M,ADR=NSSSSSSSS,FORMAT=!O!OF!D!DF!X!XF!.
DISPLAY MAIN STORE MEMORY STARTING AT ADR.     (O=>OCTAL,D=>DECIMAL,X=>HEX)
0,0.        F FOR FULLWORD (36-BIT) FORMAT.
MSDIS.      IDENTICAL TO DISPLAYMS.


MAPPROGSPACE,LEVEL=!1!2!3!,PRINT=!YES!NO!,STOP AT=NNNNNN.
DISPLAY AND PRINT MEMORY MAP
1,YES,0.
MAP.


MAPSTACK,PRINT=!YES!NO!,STOP AT=NNNNNN.
DISPLAY AND PRINT STACK MAP.
YES,0.
STKMAP.


MBP,FUNC=!SET!INI!DIS!CLEAR!,1ST=777777,2ND=777777,3RD=777777.
MICRO BREAKPOINT FUNCTIONS
SET,777777,777777,777777.
$MBP,LOCKED.


MBPPRO,ECB NO=SS.
PROCEED FROM MICRO BREAKPOINT.
1.
$MBPPR,LOCKED.


PATCH-CALC.
CALCULATOR FOR EASY MACHINE CODE PATCHES

PACALC.


PRINT,FILE=SSSSSSSSS,SCAN=!ON!OFF!.
PRINT CARD IMAGES ON PRINTER, SCAN FOR /+EJECT+/ ,ETC.
CDR,OFF. (SCAN INCLUDES LINE NUMBERS AS WOULD BE GENERATED BY THE COMPILER.)
PRSNT.


PRINTCS,FROM=NSSSSSSSS,TO=NSSSSSSSS,FORMAT=!O!OF!X!XF!D!DF!.
DUMP CONTROL STORE MEMORY TO PRINTER
0,0,0.
CSDMP.


PRINTMS,FROM=NSSSSSSSS,TO=NSSSSSSSS,FORMAT=!O!OF!D!DF!X!XF!.
DUMP MAIN STORE MEMORY TO PRINTER
0,0,0.
MSDMP.
```

```
Q-CONNECT,TO SON=!1!2!3!4!5!6!7!.
CONNECT DEBUGGER TO A SON (1-7) OR PARENT(0)
0.
QCONCT.


REGISTERS,ADR=777777,TITLE FMT=!E!I!D!T!.
DISPLAY LOCAL STORE
0, .     DEFAULT IS EASY REGISTERS, TASK FORMAT
EZREGS.


REGISTERSET,ADR=777777,REGISTER=77,VALUE=777777,TITLE FMT=!E!I!D!T!.
SET LOCAL STORE REGISTER
0,0,0, .     DEFAULT IS TASK FORMAT TITLES
SETREG,LOCKED.


SC,ADR=NSSSSSSSS,VALS=NSSSSSSSSSSSSSSSSSSS.
SET CONTROL STORE LOCATIONS FOLLOWING SPECIFIED ADDRESS.
0,0.   VALUES SEPARATED BY A /
MCSTOR.     IDENTICAL TO CHANGECSM.


SCHEMA,FILE=SSSSSSSSSSSSSSSSSSS.
SELECT SCHEMA FILE FOR SYMBOLIC DEBUG.
SCHEMA.
SCHEMA.


SM,ADR=NSSSSSSSS,VALS=NSSSSSSSSSSSSSSSSSSS.
SET MAIN STORE LOCATIONS FOLLOWING SPECIFIED ADDRESS.
0,0.   VALUES SEPARATED BY A /
MMSTOR.    IDENTICAL TO CHANGEMSM.


SOURCE,FILE=SSSSSSSSSS.
CHANGE SOURCE FILE USED IN SYMBOLIC DEBUGGING (I.E., LINE COMMAND, ETC.)
SOURCE.
NOTI. (NOT YET IMPLEMENTED)


STEP,PROC NAME=ANNNNN.
SET EASY STEPPING CONTROL.  VALID PROC NAME STEPS THAT PROC.
*.      DEFAULT TURNS STEPPING OFF, PROC NAME '$$$' STEPS EVERY PROC.
STEPEZ.


STRING,ADDRESS=NSSSSSSSS.
DISPLAY SIMPL-Q STRING VARIABLE
1.
STRDIS.
```

```
TRACEBACK,LEVEL=!1!2!3!4!.
INVOKE TRACEBACK.
1. 1=>TRACE  2=>1+LOCALS  3=>2+GLOBALS  4=>3+PROGRAM
$TRCAL.


UNLOCK,PASSWORD=SSSSSSSSSS.
UNLOCK KEYBOARD TO CHANGE MEMORY
ESP.
UNLOCK.


XREF,SYMBOL=ANNNNNNNNNNNNNNNNNN,IN PROC=ANNNNNNNNNNNNNNNNNN.
ACCESS SCHEMA FOR CROSS REFERENCE INFO AND DISPLAY.
JOE,.
XREF.
```

APPENDIX D

DISK FILE STRUCTURE ON THE QM-1

DISK FILE STRUCTURE ON THE QM-1

Disk files on the QM-1 computer are addressed by specifying the following parameters:

1. Drive No. - Selects the physical drive on the addressed control unit (0-3).

2. Cylinder No. - Selects the real cylinder for access positioning ($0\text{-}405_{10}$).

3. Head No. - Selects the desired surface on the disk. (0+1) for small (9750) disk unit. ($0\text{-}19_{10}$) for large (9755) disk unit.

4. Sector No. - There are 24 (numbered 0-23) sectors per track. (A track is one surface of a cylinder, and is specified by the cylinder # and head #).

These are the actual physical parameters used to address the desired area on the disk.

The QM-1's NOVA Emulator disk access routines (*FILES* utility) emulate the NOVA system disk access routines. Three parameters are used to specify a disk file using the NOVA system. Parameters one, two, and three are user, cylinder, and sector, respectively. These parameters, along with how they relate to the physical parameters, are discussed as follows:

1. User - A user consists of 800 tracks (19200 physical sectors), there are 2(0+1) users on the small (9750) disk units and 10(0-9) users on the large (9755) disk units.

2. Cylinder No. - A NOVA cylinder is different from a physical cylinder. A NOVA cylinder consists of two tracks (48 physical sectors). There are 400 NOVA cylinders per NOVA user.

3. Sector No. - A NOVA sector consists of two physical sectors. There are 24 NOVA sectors per NOVA cylinder.

The NOVA access routines assume octal values as input from the user. These values are then mapped to the actual physical address. Anyone doing file access on a systems level should be familiar with both forms of disk access.

An algorithm for converting from NOVA format to physical format is given as follows:

```
CYL1: = NOVA Cyl/10                          Phys Cylinder Offset into User
SEC1: = ((NOVA Cyl .mod. 10)* 48) +
        (NOVA Sec * 2)                       Phys Sector Offset into User
HS1 : = SEC1/480                             Full Cylinders from SEC1
HS2 : = SEC1 .mod. 480                       Remainder from SEC1

Head: = HS2/24                               Full Tracks from HS2
Sector=HS2 .mod. 24                          Remainder from HS2
Cylinder=CYL1 + HS1 + (User * 40 + 4)        Physical Cylinder Address
```

DISK SPACE ALLOCATION

Disk space for the EASY system is preallocated using the Nanodata NOVA Emulator System.

BASIC 9755 CAPACITIES

```
      10 NOVA user spaces per disk
     400 NOVA cylinders per user
      24 NOVA sectors per NOVA cylinder
   9,600 NOVA sectors per user
     256 QM-1 words per NOVA sector
   6,144 QM-1 words per NOVA cylinder (Note:  6144 = 14000(8) or
         about 70 80-character SIMPL-Q strings)
 2,457,600 QM-1 words per user
24,576,000 QM-1 words per 9755
```

APPENDIX E

TRIEM DEBUG SUPPLEMENT

TRIEM DEBUG SUPPLEMENT

In order to assist the programmer in debugging the TRIDENT emulation, a feature has been added to the EASY debugging system to permit TRIEM addressing. Since each TRIEM word occupies a QM-1 word pair, a given TRIDENT address need only be multiplied by two (or left shifted one) to arrive at the appropriate QM-1 address. The user may specify that the debug facility is to perform the conversion automatically by suffixing an asterisk (*) to the TRIEM address. The asterisk should be placed immediately before or in place of the terminating apostrophe as indicated below.

X'FFFF*'=X'FFFF*=X'1FFFF'

The asterisk may be used with octal and decimal addresses also.

O'1000*=D'512*=2000=D'1024

APPENDIX F

EXPORT/IMPORT USER'S GUIDE

EXPORT/IMPORT USER'S GUIDE

A package is available to enable the SIMPL-Q programmer to compile programs on the CDC 6700 and ship the binaries directly to the QM-1 via the 200UT phone link, eliminating the use of tapes. On the CDC end, there is SCOPE 3.4, INTERCOM, and an EXPORT program. On the QM-1 end, there is the 200UT Emulator, the EASY system, and an IMPORT program running under EASY.

The user should bring up the 200UT on the QM-1 and LOGIN under INTERCOM. All SIMPL-Q programs must be compiled and their binaries available (local or attached permanent files).

Type in the "CONTIN" command to identify the terminal as a remote batch terminal. Then, make sure the print queue is empty by using "H,O." and "Q,id." (where id is the two-character INTERCOM id). One job named ididIid should be in the execute queue; others in the print queue must be eliminated (see Queue Resident-File Control Commander in the CDC INTERCOM Reference Manual).

Now FETCH,EXPORT,N68 and run EXPORT by typing EXPORT,id,,,binfile." Default binfile is QM1. The id is to be specified if the EXPORTed file is to be automatically batched to the print queue as 'ididIid'. Otherwise, type EXPORT,,,, binfile,expfile. This will leave the exported file as a local file named expfile. Default expfile name is QMSHIP.

To "print" the export file on QM-1 disk, hit the Escape key and type "PD", then a Carriage Return. This tells the 200UT to print onto the disk. Enter the ON command and the file will be shipped to the QM-1. When the equipment status display comes up, the transmission is finished. Enter the OFF and LOGOUT commands.

Now that the exported file is safely on the QM-1 disk, it is time to bring it over to EASY. Bring up the EASY system on the QM-1 and type the IMPORT command (see Section V.22). The only parameter is the desired file name. The IMPORT program will restore the file to its original form and write it to the file named. The file is now ready to use.

APPENDIX G

CEXPORT/CIMPORT USER'S GUIDE

CEXPORT/CIMPORT USER'S GUIDE

A package is available to enable the programmer to ship card-image files on the CDC 6700 to the QM-1 via the 200UT link. This operates in the same manner as EXPORT/IMPORT which ships binary files (Appendix F).

The user should bring up the 200UT on the QM-1, log in under INTERCOM, and prepare the card-image file.

Type in the "CONTIN" command to identify the terminal as a remote batch terminal. Then make sure the print queue is empty by using "H,O." and "Q,id." (where id is the two-character INTERCOM id). One job named ididIid should be in the execute queue; others in the print queue must be eliminated (see Queue Resident-File Control commands in the CDC INTERCOM Reference Manual).

The CEXPORT program resides on CDCLIB, ID=N68. It does not rewind or route any files, and operator messages are written at the beginning and end of the ship file. CEXPORT is invoked by:

CEXPORT, id, source file, id, ship file.

Default source file is INPUT, default ship file is QMSHIP. The ship file must be routed by the user. Example:

```
FETCH,SOURCE,NV9.
FETCH,CDCLIB,N68.
LIBRARY,CDCLIB.
CEXPORT,,SOURCE.
ROUTE,QMSHIP,DC=PR,TID=68,FID=SHIP1.
```

The shipped file will be in the remote output queue and must be printed to disk. To do this, hit the Escape key and type "PD", then a Carriage Return. Next, enter the ON command and the file will be shipped to the QM-1. When the equipment status display comes up, the transmission is finished. Enter the OFF and LOGOUT commands.

Now that the file is on the QM-1 disk, the EASY system must be brought up and the CIMPORT command must be used (see Section V.4).

APPENDIX H

NOVA EMULATOR

NOVA EMULATOR

The NOVA Emulator is a standalone emulator running on the QM-1. The Files Utility of the NOVA Emulator is used for disk file allocation and maintenance.

A description of the initialization procedure for the NOVA Emulator Files Utility follows. It should be noted that a similar capability is provided by the FILES command under EASY.

H.1 INITIALIZATION OF THE NOVA EMULATOR FILES UTILITY (NCS) TO ALLOCATE DISK FILES

1. Set Disk Loader switch up.

2. Set the F switches down; turn the console control panel selector to IML and press the MC and START buttons.

3. Type NOV<CR> for drive 0 NOVA or NOV1<CR> for drive 1 NOVA.

4. After getting the prompt "!", type in command:

USER,n

where n is the NOVA User (0-9) you are going to specify (see Appendix D - Disk File Structure on the QM-1). The prompt "!" will be received again; reply with the command:

FILES

The system will reply with some information and then prompt for date. After entering the date, the Files Utility is ready to accept commands. All commands are single letter entries with FILES filling in the suffix letters.

The ESC key terminates the full operation last entered. When using the R command to display the directory, the information will move up the screen quickly. The display can be stopped with the space bar. To continue, use the space bar again.

Commands:

V Volume - Enter volume number (i.e., user no.) - only user 0 and the user no. specified on the USER command can be accessed.

R Report - Display directory on console.

P Print - Print directory on printer.

A Add - Add a new file.

D Delete - Delete an existing file.

E End - Terminate FILES program.

Subcommands under A or C:

A Filename ADR= - Enter address of disk location (octal format); cyl, sect, length (with space between each one). Where length is in 256 word sectors. There are 12_{10} sectors per track.

T TYPE - Type of file (any meaningful combination):
R Read only (write protected)
W Writeable
P Program in absolute image
D Data file
E Executable if type P
N Not executable
: End of type parameters

P PROGR= - Program file specification, describes absolute core image maintained on file. Three subparameters in octal format; load point, program length, entry point. All are absolute values, in words, separated by a blank.

G Go - Above entries are done. File description or alteration is completely specified. At this time the File Control Block will be displayed along with any error messages. File overlapping is allowed but produces a warning message. To complete file entry or alteration enter a carriage return. If a mistake is observed, additional subcommands may be entered following a : character. If the entire command is to be discontinued, enter an ESC code.

Subcommand under D:

Y - After display of file control block. Required to complete deletion.

When allocating space for a new file, study the directory to find available space and then figure the octal address of the location needed for the address parameter of the ADD command.

APPENDIX I

CIO

CIO is a general-purpose disk I/O package designed to support sequential and random access file I/O. Standard SIMPL-Q I/O support provides the interface to CIO and there are only limited applications whereby direct calls to CIO are desirable. The following provides the information required for calling CIO directly. Extreme care should be exercised when bypassing standard I/O procedures, and the caller should possess a thorough understanding of how CIO performs file manipulation.

$CIO - (<Request Code>, <FET Address>. <Recall Type>) - The driver routine for all of CIO. Parameters for this function are INT "REQUEST CODE", FET Address, INT "RECALL TYPE". For overlapped I/O the recall indicator should be set for "no recall." In this environment the caller is responsible for ensuring completion of the last request before initiating another. Using "automatic recall" the requested operation will be complete before the caller's next sequential instruction. CIO cannot be run at any interrupt level higher than the FNT's ECB. To do so would inhibit the lockout mechanism for permanent file directory manipulation and compromise the EASY system's integrity. See the accompanying pages for the request codes, return codes, and table formats.

$CIOINIT - This procedure must be invoked prior to use of CIO (normally $YSTEM) and should never be called by a user program. The FNTs are zeroed and CIO's critical region handler armed. If this routine should be called during normal running, the integrity of any files that are open and have been modified will be violated.

$WAEXP - If the word addressable bit of the FET is on and the file has been opened, the end of data indicator in the FNT is modified to the maximum possible size. If the file is ever written to, this change is reflected in the PFD entry. If the two conditions are not satisfied, a system error is generated (INT ARRAY FET).

LOCFILE - Provides the disk address of a file and the available size (in words). If called immediately after opening or sequential rewind, this size is that of the entire file (i.e., to end of data). For the meaning of these return values in another environment, see CIO code. To determine the total size of a file (i.e., to end of extent) LOCFILE may be preceded by a $WAEXP. (INT ARRAY FET, REFINT DAP, REF INT SIZE).

FILE ENVIRONMENT TABLE

WORD #	BITS	DESCRIPTION
0	35,36	FILE NAME AS CONTAINED IN PERMANENT FILE DIRECTORY (PFD),
1	35,36	8-BIT ASCII, 2 HIGH ORDER BITS / QM-1 WORD
2	35,18	WILL BE ZERO ...
	17,13	R/A RELATIVE SECTOR (128 QM-1 WORD PHYSICAL SECTORS)
		1ST SECTOR IS ZERO, LAST IS NOVA FILE LENGTH*2-1
	3, 1	R/A BIT, ON = >OPERATION IS R/A, ELSE SEQUENTIAL
	2, 3	CIO CODE (SEE WRITE-UP) TO REQUEST AN OPERATION
3	35,18	FOR R/A, BUFFER FIRST WORD ADDRESS (ABSOLUTE QM-1)
		FOR SEQUENTIAL, 'FIRST' OF CIRCULAR BUFFER
		WILL BE A VIRTUAL SIMPL-Q / EASY ADDRESS AT OPEN
		THAT WILL BE CONVERTED TO ABSOLUTE QM-1 AFTERWARDS
	17,18	BUFFER LAST WORD ADDRESS+1, TYPE AS FOR 'FIRST'
		ABOVE, THIS IS CIRCULAR BUFFER'S 'LIMIT'
4	35,18	CIRCULAR BUFFER'S 'IN' POINTER
	17,18	CIRCULAR BUFFER'S 'OUT' POINTER
5	35,18	WORD COUNT, DETERMINED BY 'CIO' LOGIC ON
		A SEQUENTIAL OPERATION, USER SPECIFIED ON R/A
	17,1	RECALL STATUS BIT (=0 = >AUTOMATIC, 1 => NO)
	16,1	COMPLETE " " (0 = >INCOMPLETE, 1=> COMPLETE)
	15,1	OPEN " " (1 = >FILE OPEN, 0=> NOT OPEN)
	14,1	WRITE PROTECT " (0 = >PROTECTED, 1=> WRITE OK)
	13,1	MODIFY STATUS " (0 = >UNMODIFIED, 1=> MODIFIED)
	12,1	WRITE " " (0 = >LAST OP. NOT A WRITE,
		LAST OPERATION (1 = >LAST OP. A WRITE)
	8, 1	WORD ADDRESSABLE BIT (USED BY 'DDT' I/O HANDLERS)
	3, 4	RETURN CODE (SEE P. I-6)
6	35,18	INDEX TO FNT ENTRY
	17,18	RESERVED FOR
7	35,36	USE BY
9	35,36	COMPILER &
10	35,36	'DDT' I/O HANDLER'S
11	35,36	

FILE NAME TABLE

WORD #	BITS	DESCRIPTION
0	35,36	FILE NAME AS CONTAINED IN PFD, 8-BIT ASCII, 2
1	35,36	CHARACTERS / QM-1 WORD, HIGH ORDER 2 BITS
2	35,18	OF EACH QM-1 WORD ARE O'S.
	17,9	SECTOR # OF FILES PFD ENTRY (0-23)
	8,9	1ST WORD # OF PFD ENTRY WITHIN SECTOR (0-127)
3	35,18	CURRENT FILE POSITION (IN RELATIVE SECTORS),
		0=> AT BEGINNING OF FILE (0->PFD LENGTH * 2)
	17,9	STATUS FIELD, BITS DESCRIBING FILES CURRENT STATE ,
	17,1	RECALL STATUS BIT (0/1, AUTOMATIC/NO)
	16,1	COMPLETE " " (0/1, BUSY/COMPLETE)
	15,1	OPEN " " (0/1, OPEN/NOT OPEN)
	14,1	*WRITE PROTECT* " " (0/1, *PROTECTED/NOT*)
	13,1	MODIFY " " (0/1, NOT/MODIFIED)
	12,1	WRITE " " (0/1, NOT A WRITE /
		(LAST OPERATION) WAS A WRITE)
4	35,18	END OF DATA RELATIVE SECTOR # => NEXT FOR READ/WRITE
	17,18	END OF DATA WORD COUNT => # GOOD WORDS IN SECTOR
		BEFORE EOD SECTOR.
5	35,18	END OF EXTENT RELATIVE SECTOR # = PFD'S LENGTH
		*2 (BECAUSE OF NOVA / QM-1 CONVERSION)
	17,18	1ST CYLINDER ADDRESS OF FILE (CALCULATED FROM USER
		NUMBER (NOVA))
6	35,18	1ST SECTOR ADDRESS OF FILE WITHIN CYLINDER ABOVE
	17,18	BACKWARDS POINTER TO 'FET' + LOC(FET)
7	35,18	DRIVE / UNIT # ON WHICH FILE RESIDES
	17,18	CYLINDER # ON DRIVE ON WHICH PFD RESIDES

Initially, there will be 10 FNTs of 8 words (SIMPL-Q 36-bit) each. Individual FNTs will be formatted as illustrated in the table above, and the user shall have direct access only via 'STATUS' command, and then only to the status bits. An 'R/A' status request will return in relative sector position of the FET the file's current position.

PERMANENT FILE DIRECTORY

WORD #	BITS	DESCRIPTION
0	35,36	UP TO 10 CHARACTER FILE NAME, TWO 8-BIT
1	35,36	ASCII CHARACTERS PER WORD WITH HIGH-ORDER TWO
2	35,18	BITS = 0. LEFT JUSTIFIED, ZERO FILLED.
	17,18	FIRST NOVA CYLINDER ADDRESS OF THE FILE.
3	35,18	FIRST NOVA SECTOR ADDRESS OF THE FILE.
	17,18	FILE LENGTH IN NOVA SECTORS.
4	35,17	FILE TYPE, SHOULD BE ZERO.
	18,1	WRITE ENABLE BIT (0 → READ ONLY).
	17,18	END OF DATA SECTOR
5	35,18	END OF DATA WORD COUNT
	17,18	DON'T CARE (LEFT FOR FUTURE USE).
6	35,36	LAST DATA FILE MODIFIED BY FILES COMMAND.
7	35,18	SIX CHARACTERS (MMDDYY) STORED LIKE FILE NAME.
7	17,18	CHECKSUM WORD COMPUTED SUCH THAT THE SUM OF ALL 18-BIT WORDS IN PFD ENTRY = 0.

The End of Data Sector field indicates the number of PHYSICAL sectors which contain valid data. The End of Data Word Count field indicates the number of 18-bit words which are valid data in the last sector of valid data. A zero in the End of Data Word Count means that the last sector of valid data is entirely full of valid data.

There is a directory for each user on the disk pack. Each directory contains 192 entries in the format desribed above, with empty (null) entries containing all zeroes. The entries in the directory are in an arbitrary order; they are not sorted by name or position. The directory for each user occupies physical track 1 of physical cylinder 0 of the user. In terms of NOVA disk addresses, this is 14 (octal) sectors starting at sector 14 (octal) of cylinder 0.

The NOVA PFD format is explained in additional detail in the NCS User's Reference Manual. See instructions for use of "FILES" utility for creation/maintenance of a NOVA file (to include write protection).

CIO REQUEST CODES

REQUEST CODE	DESCRIPTION
0	INVALID (CAUSES INSTRUCTION TO ABORT & RETURN CODE TO INDICATE NOT A VALID REQUEST)

1	OPEN FILE, GENERATE FNT
	RANDOM ACCESS => READ MASTER INDEX INTO SPECIFIED BUFFER (BEGINNING AT FET(3){35,18}) THE # WORDS SPECIFIED BY FET(5){35,18} OR TO END OF EXTENT.
	SEQUENTIAL => CONVERT 'FIRST' & 'LIMIT' TO ABSOLUTE ADDRESS & DEFINE 'IN' & 'OUT'.

2	REWIND, INCLUDES BUFFER FLUSHING FOR SEQUENTIAL FILES (IF LAST OP. A WRITE).
3	READ, MANAGES CIRCULAR POINTERS FOR SEQUENTIAL, ELSE READS SECTOR SPECIFIED.
4	WRITE, MANAGES CIO POINTERS ON SEQUENTIAL CALL, ALWAYS WRITES AT END OF DATA. RETURNS IN RELATIVE SECTOR AREA OF FET THE SECTOR # AT WHICH WRITE BEGAN ON R/A.
5	CLOSE, CLEARS FNT ENTRY, DEACTIVATES FILE. R/A CAUSES MASTER INDEX TO BE WRITTEN AFTER EOD. SEQUENTIAL INCLUDES BUFFER FLUSHING IF LAST OPERATION WAS A WRITE.
6	STATUS, ALWAYS WITH AUTOMATIC RECALL, RETURNS IMMEDIATELY WITH COPY OF FNT'S STATUS BITS IN FET'S STATUS FIELD.
7	REWRITE, VALID ONLY IF R/A, ELSE INVALID REQUEST FLAGGED. THIS OPTION IS VERY POWERFUL & SHOULD ONLY BE USED WHEN CALLER IS ABSOLUTELY CERTAIN OF HIS PARAMETERS. WRITES WILL NOT BE ALLOWED TO GO PAST EOD, BUT CAN & WILL GO ANYWHERE ELSE SPECIFIED BY CALLER FOR AS MANY WORDS AS REQUESTED.

CIO RETURN CODES

CODE #	DESCRIPTION
0	NORMAL TERMINATION, NO UNUSUAL CONDITIONS EXIST.
1	INVALID REQUEST CODE, REQUEST NOT SUPPORTED.
2	FILE ALREADY OPENED, ATTEMPT TO REOPEN IGNORED.
3	NO FNT SLOT AVAILABLE, OPEN ABORTED BECAUSE FNT'S FULL (CAN BE RETRIED).
4	FILE DOES NOT EXIST, REQUEST WAS MADE ON A FILE THAT HAS NO FNT ASSIGNED. IF NO FNT EXISTS THEN THE FILE IS NOT OPEN.
5	*** NOT CURRENTLY DEFINED ***
6	*** NOT CURRENTLY DEFINED *** AVAILABLE FOR FURTHER EXPANSION.
7	END OF DATA ENCOUNTERED ON A READ OR REWRITE OR END OF EXTENT ENCOUNTERED ON A WRITE.
8	DISK ERROR, HARDWARE ERROR DETECTED (IF FOUND ON OPENING, THE ERROR CODE WILL BE RETURNED IN THE FNT POINTER AREA OF THE FET).
9	ATTEMPT TO VIOLATE WRITE PROTECTION, REQUEST ABORTED.
10	PREVIOUS OPERATION INCOMPLETE, THIS REQUEST ABORTED & FET'S STATUS BITS & RETURN CODE FIELDS WILL BE "SHAKY" AT BEST FROM NOW ON. RESULTS FROM NOT CHECKING FET COMPLETE BIT WHEN WORKING WITH NO RECALL.
11	WRITE ATTEMPTED AFTER READ (INVALID ON SEQUENTIAL I/O)
12	*** NOT CURRENTLY DEFINED ***
13	'IN', 'OUT', 'FIRST', OR 'LIMIT' ERROR (I.E., SOMETHING DOES NOT COMPUTE (E.G., FIRST>= LIMIT, ETC.))
14	READ ATTEMPTED AFTER WRITE (APPLIES TO SEQUENTIAL ONLY)
15	*** NOT CURRENTLY DEFINED ***

APPENDIX J

CHARACTER SET

NANODATA QM-1 ASCII INTERNAL AND CARD CODE

Octal Code	Character	Card Code	Octal Code	Character	Card Code
000	NUL		040	SP	blank
001	SOH		041	!	12-7-8
002	STX		042	"	7-8
003	ETX		043	#	3-8
004	EOT		044	$	11-3-8
005	ENQ		045	%	
006	ACK		046	&	12
007	BEL		047	'	5-8
010	BS		050	(	12-5-8
011	HT		051	)	11-5-8
012	LF		052	*	11-4-8
013	VT		053	+	12-6-8
014	FF		054	'	0-3-8
015	CR		055	-	11
016	SO		056	.	12-3-8
017	SI		057	/	0-1
020	DLE		060	0	0
021	DC1		061	1	1
022	DC2		062	2	2
023	DC3		063	3	3
024	DC4		064	4	4
025	NAK		065	5	5
026	SYN		066	6	6
027	ETB		067	7	7
030	CAN		070	8	8
031	EM		071	9	9
032	SUB		072	:	2-8
033	ESC		073	;	11-6-8
034	FS		074	<	12-4-8
035	GS		075	=	6-8
036	RS		076	>	0-6-8
037	US		077	?	0-7-8

Octal Code	Character	Card Code	Octal Code	Character	Card Code
100	@	4-8	140		
101	A	12-1	141	a	
102	B	12-2	142	b	
103	C	12-3	143	c	
104	D	12-4	144	d	
105	E	12-5	145	e	
106	F	12-6	146	f	
107	G	12-7	147	g	
110	H	12-8	150	h	
111	I	12-9	151	i	
112	J	11-1	152	j	
113	K	11-2	153	k	
114	L	11-3	154	l	
115	M	11-4	155	m	
116	N	11-5	156	n	
117	O	11-6	157	o	
120	p	11-7	160	p	
121	Q	11-8	161	q	
122	R	11-9	162	r	
123	S	0-2	163	s	
124	T	0-3	164	t	
125	U	0-4	165	u	
126	V	0-5	166	v	
127	W	0-6	167	w	
130	X	0-7	170	x	
131	Y	0-8	171	y	
132	Z	0-9	172	z	
133	{	12-2-8	173	{	
134	/	0-2-8	174	|	
135	]	11-2-8	175	}	
136	^		176	~	
137	_		177	DEL	

Notes: 1. On IBM 029 EBCDIC keypunches, four characters are different:

OM-1 Character	IBM Character
[	¢
]	!
!	| (vertical bar)
\	0-2-8 (upper case 'T')

2. Four \ (0-2-8) in Card Columns 1-4 is an EOF.

APPENDIX K

SIMPL-Q COMPILER-IMPLEMENTED INTRINSICS

SIMPL-Q COMPILER-IMPLEMENTED INTRINSICS

The following is a list of the compiler-implemented intrinsics:

ABORT (maps into SYS$ERROR) - User abort with error #
ENDFILE - generates end-of-file indicator
READ - read values from job stream input
READC - read entire input record into string variable
READF - read item from file
REWIND - return to beginning of file
WRITE - print values of expressions
WRITEF - write item to a file
WRITEL - write record output (string)
$EASY - create EASY instruction in-line
$LINK - modify address within module of an external variable
$LINKNAME - change name of an external
$RACLS - close random access
$RAR - random access file read
$RAOPN - open mass storage file as a random access file
$RAS - random access STINDX (select different array as file index)
$RAW - write from central memory to random access file
$WACLS - close word addressable file
$WAGET - read word addressable file into specified item
$WAOPN - open file as a word addressable file
$WAPUT - write data from item into the file
EOI - test for end of information
EOIC - test for any more input records
EOIF - test for end of file
INTVAL - return decimal integer value of binary ASCII encoding of character argument
LENGTH - return length of value of argument
$MEMORY - return contents of EASY memory location
CHARF - argument converted to character
CHARVAL - converts decimal to binary ASCII encoded character (opposite of INTVAL)

The following is a list of microcode-implemented intrinsics:

Number	Mnemonic	Descriptions
1	TIME (SYSTIME)	read system clock
2	LOC	return location of item
3	MULTIPLY	extended precision multiply
4	DIVIDE	extended precision divide
5	RESTART	restart previously active PROC
6	PSLOAD	add module to program space
7	PSFREE	release modules from program space
10	COPYST	copy string to top of stack
11	CONVST (CONVSTOLD)	convert data to string
12	PACK	pack character array into string
13	UNPACK	unpack string into character array
14	SASSG	string assignment
15	CONN (CON)	string concatenate
16	FSS	fetch sub-string
17	FSSR	fetch sub-string right
20	SSS	store sub-string
21	SSSR	store sub-string right
22	MATCH	search string for match
23	SEQ	compare strings for equal
24	SNE	compare strings for not equal
25	SLT	compare strings for less than
26	SLE	compare strings for less or equal
27	SGT	compare strings for greater than
30	SGE	compare strings for greater or equal
31	TRIM	delete trailing blanks from string
32	INTFS	convert decimal string to integer
33	INTFSB	convert arbitrary string to integer
34	STRINGI	convert integer to decimal string
35	STRINGB (STRNGB8)	convert integer to arbitrary string
36	LETTER	test for letter
37	LETTERS	test string for letters
40	DIGIT	test for digit
41	DIGITS	test string for digits
42	STRNGB2	convert to string
43	STRNGB16	convert to string
44	DSHIFTL (SHFTDL)	shift left
45	DADD (ADD72)	double add
46	DSUB (SUB72)	double subtract
47	LIBRARY	set new library search order
50	MIDISARM	master interrupt disarm
51	MIARM	master interrupt arm
52	ALLOCATE	dynamically allocated external data structures
53	CONVST	convert data to string

The following is a list of the system functions:

Function Code	SYSTEM 'Function' Mnemonic	Description
0	SYS$ERROR	User abort with error number
1	SYS$MREAD	Main store transfer read
2	SYS$MSWRITE	Main store transfer write
3	SYS$CSREAD	Block transfer array from control store
4	SYS$CSWRITE	Block transfer array to control store
7	SYS$TIMESET	Set system clock
10	SYS$MSMOVE	Actual main store transfer
11	SYS$BREAK	Set breakpoint line
13	SYS$MSBYTE	Byte mode main store transfer
14	SYS$RECALL	Pass program specified recall number
15	ECB$VRST	Reset virtual device assignments
16	ECB$VDEV	Declare virtual device
17	ECB$HIO	Move TIOCB from main store to ECB
20	ECB$SIO	Move TIOCB from main store to ECB
21	ECB$SIOD	Store I/O from disk
22	ECB$WAIT	Wait on single ECB
23	ECB$WAITL	Wait on list of ECBs
24	ECB$ARM	Arm ECB, associate interrupt
25	ECB$DISARM	Disarm ECB
26	ECB$ARMC	Arm for compile
27	ECB$ARME	Arm for error
30	ECB$GET	Read ECB into array
31	ECB$PUT	Write ECB from array
32	SYS$TCPTREE	Read TID tree for user
33	SYS$TCPTCB	Read TCB for user
34	ECB$ITASK	Initialize subtask
35	ECB$STASK	Start subtask
36	ECB$KTASK	Kill subtask

APPENDIX L

AN/UYK-7 EMULATION

AN/UYK-7 EMULATION

The AN/UYK-7 emulator runs under CPX and consists of two processors: the Central Processing Unit (CPU), and the I/O processor. The AN/UYK-7 emulator is capable of handling programs as large as $128K_{10}$ and provides almost the total capability of the AN/UYK-7. In addition, debug capabilities not available on the AN/UYK-7, including pseudo-breakpoint which allows up to eight breakpoints, are a feature of the emulated AN/UYK-7. One of the few features not emulated on the AN/UYK-7 is the Externally Specified Address (ESA) word transfer ability.

The AN/UYK-7 emulator may call on up to 16 channels to associated peripherals, AN/UYK-7, or AN/UYK-20 emulators. Peripherals/emulators used in testing have included the AN/UYK-20, QRU display console, RD358 tape drive, 1004 card processor/line printer, and 1532 teletype. AN/UYK-7 programs tested in the emulated environment have included: PAM (an AN/UYK-7 debug program), Mk 92 SG (a Fire Control System Generator), a MAGIS type data base management program, and the AN/UYK-7 CMS-2 compiler system.

The desired AN/UYK-7 system with associated peripherals is built using commands under CPX including path handler (see Section IV). An entire AN/UYK-7 system configuration may be saved or recalled at any point in time using the CPX commands CHECKPOINT and RESTART (see Section IV).

The purpose of this section is to describe the two AN/UYK-7 processors. The one major difference between the real and emulated AN/UYK-7 machines is that on the real AN/UYK-7, these processors run in parallel whereas on the emulated AN/UYK-7 the processors run time sliced. Thus, there is not a one-to-one correspondence between the CPU/IOC processing ratio as done on the real AN/UYK-7s. It is possible that a program may work in one environment but not the other. This is especially true where assumptions about timing were made by the programmer and checks to prevent violations of the assumptions were not coded. This was the case, for example, in the SG Mk 92 program. It assumed some CPU instructions would be executed prior to completion of the I/O. In other programs, such as the AN/UYK-7 compiler system, no such problems were encountered.

L.1 AN/UYK-7 CPU PROCESSOR

The command file for the AN/UYK-7 CPU is CMDUYK7. This file contains the set of commands for debug, initialization, and execution of the AN/UYK-7. From the CPX level, the AN/UYK-7 emulator CPU set is invoked by entering EFUYK-7. The first part of this section describes each of the AN/UYK-7 CPU commands. The end of this section outlines the typical use of these commands.

<+>.
DISPLAY NEXT OCTAL 100 MEMORY WORDS
NO DEFAULTS
UYK7PL.

<->.
DISPLAY PREVIOUS OCTAL 100 MEMORY WORDS
NO DEFAULTS
UYK7M1.

These two commands are associated with the memory display command and will display in octal the next 100 memory words forward (+) or backward (-) from the previous memory display. This allows the operator to sequence through consecutive memory over a large program space.

<@>.
LEAVE THE CURRENT COMMAND FILE & RETURN TO PARENT.
NO DEFAULTS....
LEAVE.

This command returns the user to the CPX level of control. Exit back to the CPX level is useful for examining paths, preparing to checkpoint, or transferring to the IOC processor command set.

<CR>.
STEP UYK-7 EMULATOR
.
STEPS7.

<SPACE>.
INVOKE THE UYK-7 STATUS DISPLAY.
. NO DEFAULTS.
SSTAT7.

STEP.
PLACE THE UYK-7 EMULATOR IN STEP MODE.
.
STEPS7.

RUN.
PLACE THE UYK-7 EMULATOR IN RUN MODE.
.
RUNS7.

These commands are used to place the AN/UYK-7 in the step or run mode and to display the status of the AN/UYK-7.

The step command, as initiated by a carriage return (CR), will execute one instruction of the AN/UYK-7 and then display the status display. If the space bar is depressed, the status display will be shown without execution of the instruction. If the AN/UYK-7 processor is in the run mode (going), step will cause the processor to switch to the step mode. A typical status display is shown in Figure L-1.

```
*********************************************************************************
*                                                                               *
* UYK-7 STATUS=RUN          DATE=10/15/80 TIME=12:13:53 INSTS EXEC=212711       *
*                                                                               *
* MODE=INT  (U)=M    A0,K0,B0,S0,00024       NEXT I=AA   A0,K0,B5,S0,00000    *
*                                                                               *
* (M)=056673 OPERAND=24560024350 (P)=1/004326 (P*)=056726 (ASR)=00207002        *
*                                                                               *
*********************************************************************************
```

Figure L-1. Status Display

The STEP command has the same effect as depressing the carriage return (CR) key when the CPU is in the Run mode.

The RUN command will place the emulator in the Run mode and is equivalent to depressing the LSO/START switch to start on the real AN/UYK-7. However, if the CPU processor for the AN/UYK-7 has not been thawed, the processor will remain frozen and no commands will be executed. When the first 07 command is executed, the machine will halt unless the I/O processor has been thawed and is going.

```
STATUS-7.
INVOKE THE UYK-7 STATUS DISPLAY.
.
$STAT7.
```

This command will also invoke the status display.

The status display shows the instruction just executed and the next one to be executed, both in octal and instruction format. The pointer P value indicates the address of the next instruction to be executed both relative (P) and absolute (P*). In addition, the ASR register is presented indicating interrupt state, lockout, compare designator status, etc. Other information presented in the status display includes mode (interrupt or task), total number of instructions executed, and QM-1 computer wall clock time.

```
CHECKPOINT,TO=SSSSSSSSSS.
CHECKPOINT UYK-7 CP EMULATOR TO DESIRED FILE
1.
CHKPOI.
```

This is a less powerful checkpoint than the checkpoint at the CPX level. Only the AN/UYK-7 main store and emulator A segment are saved.

```
RESTART,FROM=SSSSSSSSSS.
RESTART THE UYK-7 CP EMULATOR FROM A DESIRED FILE
1.
RESTRT.
```

This command is used in association with CHECKPOINT to restart from a previously stored CHECKPOINT configuration.

NOTE: For a more powerful CHECKPOINT/RESTART capability, see Section IV.

```
PSEUDO-BRKPTS,=!ENABLED!DISABLED!.
ENABLE/DISABLE PSEUDO BREAKPOINTS.
=ENABLED.
PSBKPT.

SET-BREAKPOINT,#=7,TYPE=!INST!OPERAND!BOTH!,ADDRESS=777777.
SET BREAKPOINT REGISTERS.
=,INVALID,0.
MODSBK.

CLEAR-BRKPT,#=7.
CLEAR BREAKPOINT REGISTERS.
=,DISABLE,0.
MODSBK.
```

This set of instructions controls the AN/UYK-7 breakpoint capability and provides more capability than exists on an actual AN/UYK-7. With the pseudo-breakpoints disabled, which is the normal case, then just one breakpoint exists similar to a real AN/UYK-7. On the real AN/UYK-7, the CPU will halt on either an instruction address match, an operand address match, or both. This is accomplished by setting bits 18 and 19 of the breakpoint register on the real AN/UYK-7. To enable one breakpoint, the emulation operator uses the set breakpoint instruction, specifies the address and the type of breakpoint. The breakpoint number is not relevant if pseudo-breakpoints are disabled. With pseudo-breakpoints enabled, up to eight breakpoints can be set by specifying the breakpoint number. This has proved very useful where, due to flag values, it is not always clear of the directin of the CPU. The breakpoints will remain enabled until individually cleared by the CLEAR-BREAKPOINT command or totally cleared by pseudo-breakpoints disabled. Hitting a breakpoint will stop the CPU but will not stop the I/O which is controlled by a separate processor. The I/O can be halted by freezing the I/O processor or putting it in the Step mode. If the CPU executes an initiate I/O or other 07 instruction such as LIM (load IOC monitor clock), the instruction will not be executed until the I/O processor has been stopped or is in Run. The breakpoint status can be displayed by using the CMR-TASK display (see Figure L-2).

```
CMR,AREA=!TASK!INTERRUPT!DWS!.
DISPLAY UIYK-7 CONTROL MEMORY
TASK. DEFAULT TO BEGINNING OF CMR
CMRDIS.
```

The CMR-TASK display shows the task mode data including the accumulator A_0-A_7 registers, the index B_0-B_7 registers, the base registers S_0-S_7, and the breakpoint register addresses. The ASR register is interpreted. Figure L-2 illustrates an example of the CMR-TASK display.

```
*************************************************************************
*                      TASK CONTROL MEMORY REGISTERS                    *
* ACCUMULATORS    000   00000000000  00000000000  00000000000  00000000000 *
*                 004   00000000000  00000000000  00000000000  00000000000 *
*                                                                       *
* UNASSIGNED      010   0000000                                         *
*                                                                       *
* INDEX           011   0 000000     0 000000     0 000000     0 000000  *
*                 014   0 000000     0 000000     0 000000     0 000000  *
*                                                                       *
* BASE            020   000000       000000       000000       000000    *
*                 024   000000       000000       000000       000000    *
*                                                                       *
* BREAKPOINT      060   0 000000                                        *
*                                                                       *
* ASR             070   B'000000100001110000000000'                     *
*                           C   SULLLSAMLB  SONLL                       *
*                           P   TPOOOSBLBO  PFETI                       *
*                           I   A/CCCESOEO  AL//M                       *
*                           D   TLKKKLE  T  ROEGI                       *
*                               EO123 L     E QET                       *
*************************************************************************
```

Figure L-2. CMR-TASK Display

The CMR Interrupt display shows the interrupt accumulator registers A_0-A_7, the interrupt index registers B_1-B_7, and the interrupt base registers S_0-S_7. It also includes the monitor clock. Figure L-3 illustrates an example of the CMR Interrupt display.

```
*************************************************************************
*                    INTERRUPT CONTROL MEMORY REGISTERS                 *
*                                                                       *
* ACCUMULATORS    100   00000007627  00000000000  00000000000  00000000000 *
*                 104   00000000022  00000000000  00000000000  00000000000 *
*                                                                       *
* MON. CLOCK      110   0177777                                         *
*                                                                       *
* INDEX           111   0 000000     0 000023     0 000000     0 000000  *
*                 114   0 000000     0 000000     1 000034     0 000006  *
*                                                                       *
* BASE            120   003720       052400       015530       027340    *
*                 124   041150       000000       000000       000000    *
*                                                                       *
*************************************************************************
```

Figure L-3. CMR Interrupt Display

The CMR-DSW display shows the Class I/Class IV ICW and DSW values. Figure L-4 illustrates an example of the CMR-DSW display.

```
*****************************************************************************
*                                                                           *
*                      CONTROL MEMORY DSW/SPR/SIR'S                         *
*                                                                           *
* DSW-CLASS I     140  0400265 ICW  0207000 ASR  0000000 ISC  0400031 P REG  *
*                                                                           *
* DSW-CLASS II    144  0400265 ICW  0207000 ASR  0000000 ISC  0400031 P REG  *
*                                                                           *
* DSW-CLASS III   150  0400225 ICW  0207000 ASR  0000000 ISC  0400031 P REG  *
*                                                                           *
* DSW-CLASS IV    154  0400265 ICW  0207000 ASR  0000000 ISC  0400031 P REG  *
*                                                                           *
* SPR'S           160   00 000000     00 000000     00 000000     00 000000  *
*                 164   00 000000     00 000000     00 000000     00 000000  *
*                                                                           *
* SIR'S           170   0000000      0000000      0000000      0000000      *
*                 174   0000000      0000000      0000000      0000000      *
*                                                                           *
*****************************************************************************
```

Figure L-4. Control Memory DSW Display

The Indirect Control Word (ICW) indicates the address via indirect addressing which contains the location to branch to when an interrupt occurs. The ASR is the value of the Active Status Register following completion of the interrupt. The ISC is the value of the interrupt. The P register indicates the location to branch to following completion of the interrupt processing.

```
EXEC,FILE=SSSSSSSSSS.
EXECUTE A FILE OF COMMANDS
*CDR.
EXEC.
```

This command allows the user to execute a file of commands. For example, once the user is at the AN/UYK-7 level, it may be necessary to set several switches, modify the values of some registers, and place the program in Run. This could be accomplished by building a file containing the modify-stop switch, modify-jump switch, modify CMR, and modify memory commands. Through use of this command, standardized procedures for initiating the program, performing debug, or running demonstrations could be stored. These files could then be modified by using the editor when changes in operation were desired.

CHECKPOINT and RESTART used in conjunction with EXEC would minimize the number of entries required to reproduce desired sequences. Execute has been used to perform AN/UYK-7 CPU and AN/UYK-7 IOC commands within the same file.

```
INSERT-INSTR,ADDR=777777,READ-ONLY=!MAIN!NDRO!,INSTR=7777777777777,RPT=999.
CHANGE MEMORY USING INSTRUCTION FORMATTING.
0,MAIN,000000000000,1.
INSRTI.

MODIFY-MEMORY,ADDR=777777,READ-ONLY=!MAIN!NDRO!,VALUE=77777777777,RPT=999.
MODIFY UYK7 MEMORY.
0,MAIN,0,1.
MODMEM.

MODIFY-CMR,ADDRESS=777,VALUE=SSSSSSSSSS.
MODIFY UYK-7 CONTROL MEMORY
0,0.
MODCM.
```

This set of commands allows the user to modify the contents of the AN/UYK-7 emulated computer. The INSERT instruction changes the contents of any address in main memory or nondestructive read-out memory (NDRO) as specified in instruction format. For example, the characters entered for format 1, define the f, a, k, b, i, s, and y fields. The number of characters left-to-right forming each field in a format 1 instruction is 2, 1, 1, 1, 1, 1, and 5. To set consecutive addresses to the same value, the repeat feature RPT is used to indicate the number of consecutive calls duplicated. The address used should be the absolute address. The user should be cautious in entering format IV and I/O instructions. Once the instruction is entered, it is displayed in the instruction pneumonics along with the seven adjacent cells in the block of 10_8 cells containing the modified instruction address.

The MODIFY-MEMORY command is similar to the MODIFY INSTRUCTION command except the value entered is a whole word in octal format. The memory display of 100 octal cells will follow this command.

The MODIFY-CMR command allows the user to change CP control memory including the A, B, and S registers, the ASR, and the ICW, CDW interrupt data.

```
INSTRUCTIONS,ADDR=777777,READ-ONLY=!MAIN!NDRO!.
DISPLAY MEMORY IN INSTRUCTION FORMAT.
0,MAIN.
INSTUY.

MEMORY,ADDR=777777,READ-ONLY=!MAIN!NDRO!.
DISPLAY UYK-7 MEMORY (MAIN & NDRO)
0,MAIN.
UYK7ME.
```

These two commands provide for display of memory in instruction or octal format. Figure L-5 shows an instruction display and Figure L-6 shows an octal display.

```
*************************************************************************************
*                    AN/UYK-7 MAIN MEMORY DISPLAY                                   *
*                                                                                   *
*     ADDR       CONTENTS      INSTR. FMT.                                          *
*   056720    25410024333   53 0 1 0 0 1 04333      JNE   A0,B0,S1,04333           *
*   056721    30066600000   60 1 5 5 1/00 0 0 0 0   HCSI  A1,B5,1/DATA             *
*   056722    22240000006   44 5 0 0 0 0 00006      C     A5,K0,B0,S0,00006        *
*   056723    25650024333   53 5 1 0 0 1 04333      JLE   A5,B0,S1,04333           *
*   056724    30060200000   60 1 4 0 1/00 0 0 0 0   HSCI  A1,B0,1/DATA             *
*   056725    20000000024   40 0 0 0 0 0 00024      M     A0,K0,B0,S0,00024        *
*   056726    06002400000   14 0 0 5 0 0 0000       AA    A0,K0,B8,S0,00000        *
*   056727    30474200000   61 1 7 0 1/00 0 0 0 0   HLCI  A1,B0,I/DATA             *
*                                                                                   *
*************************************************************************************
```

Figure L-5. Instruction Display

```
*************************************************************************************
*                          AN/UYK-7 MAIN MEMORY DISPLAY                             *
*                                                                                   *
*   056700   30066600000   22240000010   25610024306   30064200000                 *
*   056704   22000000012   25650024315   30060600000   20040000024                 *
*   056710   06042400000   30470600000   04154000455   12157000471                 *
*   056714   25430024350   10330070106   30056200000   22233107616                 *
*   056720   25410024333   30066600000   22240000006   25650024333                 *
*   056724   30060200000   20000000024   06002400000   30474200000                 *
*   056730   04114000455   12117400471   25430024350   10374070106                 *
*   056734   30055600000   22177507616   25450024350   30056200000                 *
*   056740   22233107616   25450024350   30060200000   20000000024                 *
*   056744   06002400000   30470200000   04114000455   12117000471                 *
*   056750   10640000001   30065600000   22140000023   25650024253                 *
*   056754   10370070106   30056200000   22233507616   25410024361                 *
*   056760   16670070106   10374070106   30054200000   22037507616                 *
*   056764   25410024366   16474070106   10540000001   30055200000                 *
*   056770   22100000023   25650024247   10440000001   30045600000                 *
*   056774   22140000011   25650024241   10154024414   10214024415                 *
*                                                                                   *
*************************************************************************************
```

Figure L-6. Memory Display

```
MASTER-CLEAR.
MASTER CLEAR SEQUENCE OF UYK7.

MASTCL.

MODIFY-BOOT-SW,SWITCH=!0/1/2!MODE!,SETTING=!0!1!2!AUTO-REC!MANUAL!NEUTRAL!.
MODIFY BOOTSTRAP SWITCHES.
0/1/2,1.
MBOOTS.

MODIFY-BRK-SW,SETTING=!PROGRAM!MANUAL!.
MODIFY BREAKPOINT SWITCH.
PROGRAM.
MBKPTS.

MODIFY-JUMP-SW,SWITCH=!1!2!3!,SETTING=!ENABLE!DISABLE!.
MODIFY JUMP SWITCHES.
1,ENABLE.
MODSJS.

MODIFY-STOP-SW,SWITCH=!5!6!7!,SETTING=!ENABLE!DISABLE!.
MODIFY STOP SWITCHES.
5,ENABLE.
MODSJS.
```

These emulator commands are used to emulate the switches on the Maintenance Unit Panel of the AN/UYK-7. The MASTER-CLEAR will clear all breakpoints, clear the program pointer P, and place the AN/UYK-7 in the interrupt state under class IV with class I, II, and III locked out. It will perform all the functions of the MASTER-CLEAR toggle switch on the AN/UYK-7 panel.

The MODIFY-BOOT-SWITCH command performs the functions of the two bootstrap toggle switches. The 0-1-2 switch is used to load bootstrap and diagnostic programs. Switch positions 1 and 2 determine which of the two available boostrap programs is loaded. The diagnostic program is selected by position 0. The auto-recovery/neutral/manual option determines the other switch setting.

The PROGRAM/MANUAL switch is used to switch to the Initial Condition Word 2 address (ICW2) if a breakpoint is detected and the switch is in PROGRAM. Otherwise, a breakpoint will generate a halt. This command emulates the PROGRAM/MANUAL switch and the response to the switch setting.

The MODIFY-JUMP switch and MODIFY-STOP switch commands emulate jump switches 1, 2, 3, and stop switches 5, 6, and 7. The AN/UYK-7 program CPU will jump or stop if the switches are detected as set by the program.

```
SWITCHES.
DISPLAY UYK7 SWITCHES.

SWITCH.
```

The command switches show the status of all switches. Figure L-7 illustrates a sample switch configuration. The state of each switch is indicated in the asterisks.

```
*********************************************************************************
*                                  *                                            *
*          BREAKPOINT              *                  JUMP                      *
*          ----------              *                  ----                      *
*                                  *          1         2         3              *
*                                  *      ---------- ---------- ----------       *
*          PROGRAM                 *       ENABLED    ENABLED    ENABLED        *
*                                  *                                            *
*         *MANUAL                  *      *DISABLED  *DISABLED  *DISABLED       *
*                                  *                                            *
*********************************************************************************
*                                  *                                            *
*          BOOTSTRAP               *                  STOP                      *
*          ---------               *                  ----                      *
*       1          AUTO REC       *          5         6         7              *
*                                  *      ---------- ---------- ----------       *
*      *0                          *       ENABLED    ENABLED    ENABLED        *
*                                  *                                            *
*       2          MANUAL          *      *DISABLED  *DISABLED  *DISABLED       *
*                                  *                                            *
*********************************************************************************
```

Figure L-7. Switch Display

```
MODIFY-P,BASE=7,DISPLACEMENT=177777.
MODIFY P REGISTER.
0,0.
MOD$P.
```

This command allows the user to modify the address of the next instruction to be executed and the base register under which the program is running. This is primarily used during program debug to check out specific areas of code.

```
PRINT,=!OFF!ON!.
TURN PRINTING ON OR OFF.
OFF.
MODPRT.
```

This command turns the printer on or off. At initialization, the print capability is off. Once the printer is turned on, all print generated to the CRT will also be generated to the printer. The line printer must be manually enabled or an error message to turn on the printer will occur. This command is only valid at this level and will not print if the user returns to a higher level without turning print off.

```
TAPE-TO-MEMORY,MT-UNIT=1,OFFSET=777777.
UYK-7 TAPE TO MEMORY.  OFFSET = MEMLIM IMPLIES NDRO.
0,0.
TPTOME.
```

This command is used to load bootstrap loadable bioctal tapes into the AN/UYK-7. To load NDRO, OFFSET = MIMLIM is used. NDRO should be loaded before running any AN/UYK-7 programs in order to get the class I, II, and III interrupts. The offset will be the initial program address to which is added any offset built into the load tape.

```
SPY,EXPR=SSSSSSSSSSSSSSSSSS,MIN=777777,MAX=777777,BINSIZE=777777,MASK=777777.
SPY ON 'EXPRESSION' DURING EXECUTION. PRINTS A HISTOGRAM OF VALUES (BY <@>).
CS(502),1000,7700,10,777777.
SPYON.
```

This command allows the user to examine control store or main store such as control store addresses 502 in the default case. All stack identifiers are also supported. The user defines the bins by MIN, MAX, and bin size. For example, for MIN = 10, MAX = 30, BINSIZE = 5, four bins of 10-15, 15-20, 20-25, and 25-30, exclusive of the upper limit. The MASK allows a portion of the sampled value to be masked off for use in the plot. The sample is taken every 100 ms. The results of SPY are printed following entry of a @. SPY is most useful for studying timing and program loading. A SPY of a flag used for a switch would show percentage of number of calls to each routine. Figure L-8 illustrates a sample of 57 data values of CS(502) showing that all samples were above or equal to 7710.

```
                    SPY RESULTS

        SAMPLE TAKEN AT ADDRESS CS(502)                    10/16/80   11:46:32

BINSIZE:0'000000000010'     MINIMUM:0'000000007700'     MAXIMUM:0'000000007710'

                                NO. IN SAMPLE:57


                                PERCENTAGE OF SAMPLE IN EACH BIN

                          0                   5                              25
                          ---------------------------------------------------------

BELOW MINIMUM             X
.GE. 0'000000007700'      X
MAXIMUM AND ABOVE         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 10
```

Figure L-8. SPY Example

AD-A100 522 NAVAL SURFACE WEAPONS CENTER DAHLGREN VA F/G 9/2
EMULATION AID SYSTEM II (EASY II) SYSTEM PROGRAMMER'S GUIDE.(U)
MAR 81 C J NAPLES
UNCLASSIFIED NSWC/TR-81-98 NL

END
DATE FILMED
7-81
DTIC

L.2 UYK-7 IOC PROCESSOR

The command file for the AN/UYK-7 IOC is CMDIOC7. This file contains the set of commands for controlling the IOC and displaying the state of the I/O. Certain commands are similar to the AN/UYK-7 CPU commands and therefore are not described again. These include @, CR, EXEC, RUN, SPY, and STEP. The procedures called are LEAVE., IOC$ST., EXEC., IOC$GO., SPYON., and IOC$ST., respectively.

```
CMR-IOC,AREA=!INPUT!OUTPUT!EXTERNAL-FUNCT!EXTERNAL-INT!.
DISPLAY IOC-7 CONTROL MEMORY.
INPUT.
IOCC MR.
```

There are three display commands in the IOC level. This command displays the latest I/O executed for input, output, external functions, and external interrupts. Each of the 16 channels is displayed. For each channel, the command address, PWD, CF, MIF, BP, FIN, buffer, and current address are displayed. The command address gives the address of the first word to I/O. Figure L-9 shows the example of an input display.

COMMAND WORD FORMAT

CHNL#	COMM. ADD. PT.	PWD	CF	MIF	BP	FIN. BUFFER	CURR. ADDR.
000000	000000	00	0	0	00	00000	000000
000001	000000	00	0	0	00	00000	000000
000002	051700	11	0	1	00	00013	027340
000003	000000	00	0	0	00	00000	000000
000004	000000	00	0	0	00	00000	000000
000005	000000	00	0	0	00	00000	000000
000006	000000	00	0	0	00	00000	000000
000007	000000	00	0	0	00	00000	000000
000010	000000	00	0	0	00	00000	000000
000011	000000	00	0	0	00	00000	000000
000012	000000	00	0	0	00	00000	000000
000013	000000	00	0	0	00	00000	000000
000014	000000	00	0	0	00	00000	000000
000015	000000	00	0	0	00	00000	000000
000016	000000	00	0	0	00	00000	000000
000017	000000	00	0	0	00	00000	000000

Figure L-9. CMR-IOC Input

In this example, an input on channel 2 exists. The partial word designator is 3 indicating a whole-word transfer. The CAP (Command Address Pointer) points to the next IOC instruction to be executed. The current instruction, an IB instruction, is located at 51677 (i.e., 51700-1). The IB instruction has a monitor interrupt flag set. The input data is to be stored starting at location 27340. Figure L-11 shows the IB instruction. BP is the byte pointer identifying the partial-word position in memory involved in the transfer. Since this is a whole word transfer, BP is zero. The FIN buffer is the final address, compare bits of the last word in the buffer. This should read 27343, not 13. This is a four-word transfer.

```
FLIP-FLOPS,=!MTR-INT!CHN-ACT!BUF-PND!BUF-ACT!.
DISPLAY IOC-7 FLIP FLOPS.
CHN-ACT.
FLIPD.
```

The FLIP-FLOP command displays the status of I/O flip-flops used in the EASY I/O system. This display and the display paths available through CPX present a clear view of the status of I/O for each channel. Figure L-10 illustrates a flip-flop display.

BUF-PND FLIP FLOPS

CHANNEL STATUS

CHANNEL NUMBER	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INPUT	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
OUTPUT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EXTERNAL FUNCTION	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EXTERNAL INTERRUPT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure L-10. Flip-Flop Display

There is a flip-flop display for buffer pending, buffer active, monitor interrupt, and chain active. In the above display, input was opened on channel 3 which resulted in a buffer pending and a buffer active. The chain active bit was clear since this is a one-word chain not initiated by an AIC. The interrupt is clear because the data transfer is not complete.

```
MEMORY-IOC,ADR=777777.
DISPLAY IOC-7 MEMORY.
0.
IOCCWD.
```

The MEMORY-IOC command will display 10 addresses in octal, actual by I/O field f, k, j, m, c, and y, and by I/O mnemonics. Figure L-11 illustrates a memory IOC display.

!!MEMORY-IOC,ADR=51677.

ADDR.	CONTENTS	MNEMONIC	F	K	INSTR. FMT. J	M	C	Y
051670	03420110531	07	0	04	0	0	1	10531
051671	03420110532	07	0	04	0	0	1	10532
051672	03420110533	07	0	04	0	0	1	10533
051673	03420110534	07	0	04	0	0	1	10534
051674	03404100010	07	0	01	0	0	1	00010
051675	05314052007	FB12	3	03	0	0	0	52007
051676	04716052011	OB11	3	03	1	0	0	52011
051677	04316052012	IB10	3	03	1	0	0	52012

Figure L-11. Memory IOC Display

Address 51677 contains the IB instruction referenced in Figure L-9. The buffer control word is contained at 52012.

```
IOC-NUMBER,=!0!1!2!3!.
DEFINE THE IOC NUMBER FOR THIS PROCESSOR.
0.
IOCNU.
```

This command allows the user to specify the IOC number. The default IOC number is zero.

```
PRINT,=!OFF!ON!.
TURN PRINTING ON OR OFF.
OFF.
MODPRT.
```

The PRINT command turns the printer on and off. This would normally allow the user to print displays just as is done by the CPU processor. However, the PRINT commands were not implemented in the IOC display processing. Therefore, this command has no effect. To obtain a print of the IOC display, the QM-1 operator can use the Control-Q entry followed by a line-feed entry which will print the data on the screen. However, this will not print out the nonstandardized characters correctly and will also print out control characters.

APPENDIX M

AN/UYK-20 EMULATION

AN/UYK-20 EMULATION

The AN/UYK-20 emulator can be used to run AN/UYK-20 jobs in a similar fashion to running the job on the real AN/UYK-20. The purpose of this section is to give the user a summary reference manual for using the AN/UYK-20 emulator. The following discussion assumes computer control is at the AN/UYK-20 level. Subsequent paragraphs instruct the user how to get to this level.

An AN/UYK-20 program can be created on the system or, if the program is on a bootstrap loadable tape, it can be read into memory. The command for reading in a tape is TAPE-TO-MEMORY. By using MASTER-CLEAR and setting the required emulated switches and registers, the initial program environment can be established. To execute the program, the user should specify the RUN command. If the user desires to execute the program one line at a time and view the registers/memory after each instruction is executed, then STEP should be used instead of RUN.

To debug the program, the user has available instruction, register switches, and memory displays. Displays for examining the I/O are also available. To stop the program at a specific location, a breakpoint can be set. The operator can set up to seven additional "pseudo" breakpoints to provide debug capability not available on the real AN/UYK-20.

Additional capabilities to perform debug are available at the CPX level such as CHECKPOINT. In order to checkpoint his system, the user should enter <@> to return to the parent (CPX) level. Then, the active processors must be frozen (FREEZE) and the state of the entire emulated configuration may be saved using CHECKPOINT. This saves all registers, memory, instructions, and the state of all associated emulators. This checkpointed configuration can then be reestablished by using RESTART and THAW.

The following summarizes how to use the AN/UYK-20 emulator:

1. Spin up the disk drive.
2. Mount program tapes.
3. Bring up EASY emulator.
    a. MC (Master Clear)
    b. Start
    c. LD EASY
    d. Date
    e. Time
    f. @.

    See EASY SPG, Appendix A

4. Select the directories required to run the AN/UYK-20 and select the AN/UYK-20 library. The commands DIRECTORY and LIBRARY are used.

5. Select the appropriate configuration using CPX or CPX with RESTART for CHECKPOINT configurations.

6. Enter UYK20CPEF to enter the AN/UYK-20 emulation program.

7. Load the program using TAPE-TO-MEMORY.

8. Initialize registers and switches to starting desired values.

9. Run the program performing debug and I/O.

The entries to operate the AN/UYK-20 emulator are described as follows:

1. UYK20CPEF calls the AN/UYK-20 initialization routines and the recall handler;

2. <+> displays the "next" 100 (octal) AN/UYK-20 memory words;

3. <-> displays the "previous" 100 (octal) AN/UYK-20 memory words;

4. <@> returns to parent level;

5. <CR> puts the AN/UYK-20 emulator into the "step" mode and displays the status display;

6. <SP> invokes the AN/UYK-20 status display;

7. BREAKPOINTS invokes the AN/UYK-20 regsiters/switches/breakpoints display;

8. CLR-ALL-BKPTS sets the read/write-breakpoint register and the seven pseudo-breakpoint registers to zero, regardless of the setting of the breakpoint switches;

9. INSTRUCTIONS displays 10 (octal) AN/UYK-20 instructions in three different formats (straight octal representation of 16 bits, octal representation of the "unassembled" instruction, and mnemonic representation of the op code in "unassembled" format);

10. MEMORY displays the "current" 100 (octal) AN/UYK-20 memory words;

11. MOD-BRK-SW turns on/off the read- and/or write-breakpoint switch(es), or enables/disables the pseudo-breakpoint switch;

12. SET-BOOT-SW allows the user to select either bootstrap switch 1 or 2;

13. MOD-BREAKPOINT modifies the read/write-breakpoint register or one of the seven pseudo-breakpoint registers if the pseudo-breakpoint switch is enabled;

14. MOD-CLOCK-SW turns on/off the real-time and/or monitor clock switches;

15. MOD-DEV-INSTR modifies AN/UYK-20 instructions of the "deviation-value" type format (i.e., local jump instructions);

16. MOD-GEN-INSTR modifies those AN/UYK-20 instructions of the "general" type format;

17. MOD-GEN-REGS modifies a register in the primay or alternate general register set, or modifies a register in the general register set in use, according to the applicable bit in Status Register 1;

18. MOD-MEMORY modifies the contents of a location in main or NDRO memory;

19. MOD-P modifies the program register;

20. MOD-STAT-REG modifies the contents of Status Register 1 or 2;

21. MOD-STOP-SW turns on/off Stop Switches 1 or 2;

22. PRINT turns on/off an internal print flag, thus controlling printing of subsequent terminal displays;

23. REGISTERS invokes the registers/switches/breakpoints display;

24. RUN puts the AN/UYK-20 into the "run" mode;

25. STATUS-REGS invokes a display of the individual bits in Status Registers 1 and 2, with an accompanying description of bit function;

26. STEP, same as 5 above;

27. STATUS-20, same as (6) above;

28. SWITCHES invokes the registers/switches/breakpoints display;

29. TAPE-TO-MEMORY transfers the contents of an AN/UYK-20 bootstrap loadable tape into the emulated machine;

30. CONTROL-MEMORY displays I/O control memory for the specified channel;

31. MOD-INT-ADDR modifies interrupt branch or storage addresses for the specified interrupt class;

32. MOD-CNTL-MEM modifies I/O control memory for the specified channel;

33. MASTER-CLEAR initializes the AN/UYK-20;

34. COMMAND-CELL displays AN/UYK-20 I/O Command Instruction at location 0140 and 0141; and

35. INTERRUPT displays the current and restore values for class 1, 2, or 3 interrupts.

Figures M-1 through M-8 are examples of AN/UYK-20 displays.

```
AN/UYK-20 MAIN MEMORY DISPLAY - CURRENT

001000   000000 105100 005515 102775 000000 105100 003666 102775   001007
001010   000000 105100 005315 102775 000000 105100 005371 102775   001017
001020   000000 105100 005210 102775 000000 105100 005463 102775   001027
001030   007547 005702 003660 005704 101200 000000 023500 001035   001037
001040   027547 005702 023660 005704 100734 000001 000002 000000   001047
001050   000000 000000 000000 000000 000000 000000 000000 000000   001057
001060   000000 000000 000000 000000 000000 000000 000000 000000   001067
001070   000000 000000 000000 000000 004516 004626 004636 004647   001077
```

Figure M-1. Example of Memory Display


```
UYK-20 STATUS=STEP   DATE=09/30/00  TIME=10:08:44   INSTS EXEC=4

      (IR)=    S 04 00 005514       NEXT I=   LJ $-25

(M)=005514 (M*)=005514 (P)=005517 (P*)=005517 SR1=000000 SR2=000000
```

```
UYK-20 STATUS=STEP   DATE=09/30/00  TIME=10:08:46   INSTS EXEC=5

      (IR)=   LJ $-25             NEXT I=   SZ 00 00 002546

(M)=005517 (M*)=005517 (P)=005472 (P*)=005472 SR1=000000 SR2=000000
```

Figure M-2. Examples of Status Display

```
UYK-20 STATUS=STEP    DATE=09/30/00  TIME=10:10:02    INSTS EXEC=6

      (IR) =   SZ 00 00 002546       NEXT I=  JLR 04 00 005515

(M)=002546 (M*)=002546 (P)=001001 (P*)=001001 SR1=000000 SR2=000000
```

```
UYK-20 STATUS=BKPT    DATE=09/30/00  TIME=10:10:06    INSTS EXEC=32

      (IR) =   SD 04 00 000141       NEXT I= IDCR 00 00

(M)=000141 (M*)=000141 (P)=005330 (P*)=005330 SR1=001400 SR2=000000
```

Figure M-2. Examples of Status Display (Cont'd)

```
              AN/UYK-20 REGISTERS/SWITCHES/BREAKPOINTS DISPLAY

  R E G I S T E R S    S W I T C H E S    B R E A K P O I N T S    C L O C K

       P   000000        BOOTSTRAP   1        READ      ON       UPPER  000000
     SR1   000000          STOP 1    OFF      WRITE     ON       LOWER  000000
     SR2   000000          STOP 2    OFF    PSEUDO    ENABLED

ALTERNATE            G E N E R A L   R E G I S T E R S            ALTERNATE

      00   000000 000000 000004 033033 023000 000000 000000 000000    07
      10   000000 000000 000077 001177 145010 000000 145111 000000    17


                          B R E A K P O I N T S

      00   001005 001777 000000 000000 000000 000000 000000 000000    07
```

Figure M-3. Examples of Registers, Switches, or Breakpoints Display

```
STATUS REGISTER 1                    STATUS REGISTER 1
BIT NO  DESCRIPTION                  BIT NO  DESCRIPTION (CONT'D)

   15=  0  NOT USED                    5&4=  00  NOT USED

   14=  0  GEN. STACK REG. DESIGNATOR     3=   0  IF 0, CLASS 1 LOCKED OUT

   13=  0  NOT USED                       2=   0  IF 0, CLASS 2 LOCKED OUT

   12=  0  NDRO MODE DESIGNATOR           1=   0  IF 0, CLASS 3 LOCKED OUT

   11=  0  CARRY DESIGNATOR               0=   0  IF 0, DISABLE DMA

   10=  0  OVERFLOW DESIGNATOR         STATUS REGISTER 2
                                       BIT NO    DESCRIPTION
  9&8= 00  CONDITION CODE
                                       15&14= 00  IF M=16 ) NORMAL IN-
    7=  0  # F.P. UNDERFLOW/OVERFLOW   13&12= 00  IF M=14 ) DIRECT
             INTERRUPT (0=ENABLED)     11&10= 00  IF M=12 ) ADDRESSING
                                        9&8=  00  IF M=10 ) INDICATOR
    6=  0  # F.P. ROUNDED INTERRUPT      7-0=  00 000 000  I/O INSTR FAULT
             (0=ENABLED)                          & MEMORY RESUME INT DATA
```

Figure M-4. Example of Status Register Display

```
AN/UYK-20 CLASS III INTERRUPT

     INTERRUPT AT          REAL-TIME CLOCK     STATUS REG.    1        2

     P-ADDRESS 000000      UPPER 176000          CONTENTS 000000  000000
  RELOAD VALUE 063716      LOWER 000000       RELOAD VALUE 162220  000000


                      AN/UYK-20 IOC COMMAND CELL

     LOC.140/141= 71 2 00 06 001240          INSTR= DCK 00 06 001240

          INSTRUCTION WAS NOT EXECUTED          CHANNEL= 00

    LOC                 EXTERNAL INTERRUPT WORD STORAGE                  LOC

      200   000000 000000 000000 000000 000000 000000 000000 000000     207
      210   000000 000000 000000 000000 000000 000000 000000 000000     220
```

Figure M-5. Example of Interrupt Display (for Class III interrupt)

```
AN/UYK-20 CONTROL MEMORY DISPLAY -- CHANNEL 00

          I N P U T          O U T P U T          S E R I A L

    TM          ABORT              ABORT          MONITOR REG 000200
  BYTE          UPPER              UPPER          SUPPRESS REG 000000
   BWC          2551               0000                 CLOCK   LO
   BAP          003124             000000           STOP-BITS   ONE
   CAP          003477             000000       PARITY-CHECKING DISABLED
  INST     LK 04 00 100000      - 00 00          SELECT-PARITY   ODD
        01 2 04 00 100000  00 0 00 00              CHARACTER  5-BIT
```

Figure M-6. Example of Control Memory Display


```
AN/UYK-20 IOC COMMAND CELL

  LOC.140/141= 71 2 00 06 001240          INSTR=  OCK 00 06 001240

        INSTRUCTION WAS NOT EXECUTED          CHANNEL= 00


  LOC                EXTERNAL INTERRUPT WORD STORAGE                LOC

   200   000000 000000 000000 000000 000000 000000 000000 000000   207
   210   000000 000000 000000 000000 000000 000000 000000 000000   220
```

Figure M-7. Example of Command Cell

AN/UYK-20 MAIN MEMORY INSTRUCTION DISPLAY

ADDRESS	CONTENTS	INSTRUCTION FORMAT	MNEUMONIC
001000	000000	00 0 00 00	- 00 00
001001	105100 005515	42 2 04 00 005515	JLR 04 00 005515
001003	102775 000000	41 1 375	LJI $-3
001005	105100 003666	42 2 04 00 003666	JLR 04 00 003666
001007	102775 000000	41 1 375	LJI $-3
001011	105100 005315	42 2 04 00 005315	JLR 04 00 005315
001013	102775 000000	41 1 375	LJI $-3
001015	105100 005371	42 2 04 00 005371	JLR 04 00 005371

Figure M-8. Example of Instruction Display

DISTRIBUTION

USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Ray, CA 90291
ATTN: Lou Gallenson

Defense & Space Systems Group of TRW Inc.
One Space Park
Redondo Beach, CA 90278
ATTN: David Bixler
Herb Wangenheim

John S. Hale
P.O. Box D5
Stony Brook, NY 11790

Dr. John Tartar
Department of Computer Science
University of Alberta
Edmonton Alberta
Canada T6G 2E1

Jon Humphry
Hughes Aircraft Company
Bldg. 12 M/S V107
Culver City, CA 90230

Commander
Naval Ocean Systems Center
271 Catalina Boulevard
San Diego, CA 92152
ATTN: Code 5200
Russ Eyers

Commanding Officer
U.S. Army Harry Diamond Laboratories
2800 Powder Mill Road
Adelphi, MD 20783
ATTN: Branch 520
Rick Johnson

Defense Documentation Center
Cameron Station
Alexandria, VA 22314 (10)

U.S. Naval Electronic Systems Command
Washington, D.C. 20360
ATTN: John Machado
(Code 330)

DISTRIBUTION (Cont'd)

Robert V. Hanzlian
224 Courtland St.
Buffalo, NY 14205

Director
U.S. Army TRADOC System Analysis
Activity
White Sands Missile Range, NM 88002
ATTN: ATAA-SL (Technical Library)

Computer Science Department
University of Maryland
College Park, MD 20742
ATTN: Dr. Yaohan Chu
Dr. Victor Basili

Paul A. Boudreaux
University of Southwestern Louisiana
USL Box 4-4330
Lafayette, LA 70504

DIT-MCO International
5612 Brighton Terrace
Kansas City, MO 64130
ATTN: J. L. Herbsman

Boeing Commercial Airplane Company
P.O. Box 3707
Seattle, WA 98124
ATTN: Keith Mitchell, M.S. 9R-08
Flight Control Lab

Hughes Aircraft Company
P.O. Box 92426
Los Angeles, CA 90009
ATTN: Steve Jackson, M.S. R1/B218

Intermetrics, Inc.
5392 Bolsa Avenue
Huntington Beach, CA 92647
ATTN: Marjorie Kirchoff

McDonnell Douglas Astronautics Co.
5301 Bolsa Avenue
Huntington Beach, CA 92647
ATTN: S. Harris Dalrymple, M.S. A3-236-10-1

Robins Air Force Base
Warner Robins, GA 31098
ATTN: John Douglas
ACL/MMECV

DISTRIBUTION (Cont'd)

USAF RADC
Griffiss AFB, NY 13441
ATTN: Armand Vito, ISCA

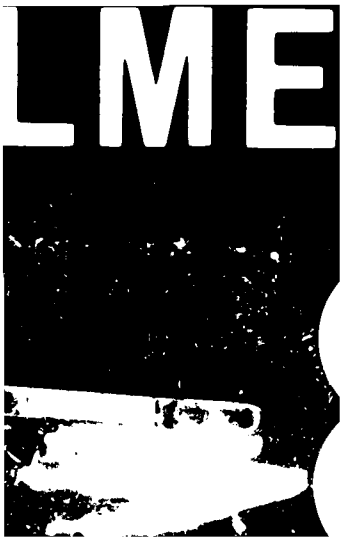
Tinker Air Logistics Center
Tinker AFB
Oklahoma City, OK 73145
ATTN: Gary M. Parish, MMECO

Nanodata Computer Corporation
One Computer Park
Buffalo, NY 14203
ATTN: Mike Senft
Bill Robertson

System Development Corporation
601 Caroline Street
Fredericksburg, VA 22401
ATTN: Paul Tiffany

Douglas Martindale
OC-AOC/MMECO
Tinker AFB
Oklahoma City, OK 73145

Local:
K33 (40)
K10
K20
K40
K50
X210 (6)


LME