DATABOARD 4680®

OS.8 MT
OPERATOR'S MANUAL

SATTCO AB DALVÄGEN 10 S-171 36 SOLNA SWEDEN PHONE 08/ 734 00 40 TLX 11588

FOREWORD

This manual contains detailed reference material needed to control an OS.8 Operating System, and deals specifically with revision 2.50 and higher revisions. The manual is intended as an operator's reference manual for the Operating System.

This publication is divided into two separate manuals and three appendices:

OS.8 TERMINAL REFERENCE MANUAL (TRM)

contains information pertinent to the Mult-Terminal Management (MTM). It is designed primarily for the terminal user, however, it could prove useful to the System programmer.

OS.8 UTILITY REFERENCE MANUAL (URM)

describes the system utility programs such as disc maintenance, back-up handling, program development and language processors.

Appendix A contains a command summary.

Appendix B holds the crash codes.

Appendix C specifies the error codes.

OS.8

TERMINAL REFERENCE MANUAL (TRM)

CONTENT

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Operating System OS.8 is a Real-Time Multi-Tasking System normally controlled from an interactive terminal. The terminal may be any that is TTY-compatible such as CRT.

Since the operator plays an important role in the successful operation, it is necessary to understand the facilities available. These facilities and the operating procedures necessary to run the Operating System are described in this manual.

The OS.8 Programming Manual should be referred to if more detailed information about the operating system is required.

The user ougth to be familiar with the following documents:
- DataBoard-4680 System Manual.
- DataBoard-4680 Software Catalog.

This manual is divided into four more chapters:

Chapter 2 describes the start-up and shut-down procedures.

Chapter 3 contains a general description of the Terminal Management.

Chapter 4 specifies the general command syntax structure.

Chapter 5 explains terminal user commands.

1.2 ENVIRONMENT

All tasks in the system are served on a priority basis. Thus, real-time applications may be executed at a high priority to ensure timely response. Task priorities are set at task load time and may be modified by the terminal operator.

1.3 FILE SYSTEM OPERATIONS

The operator frequently interacts with the Disk File System through terminal commands. These commands are described in Chapter 5 of this manual.

1.4 SYSTEM CRASHES

When the system determines that futher execution may cause system or user data to be destroyed, the system crashes in a controlled way. The Operating System MUST be reloaded after a crash ! Refer to DataBoard-4680 OS.8 Programming Manual for more detailed information.

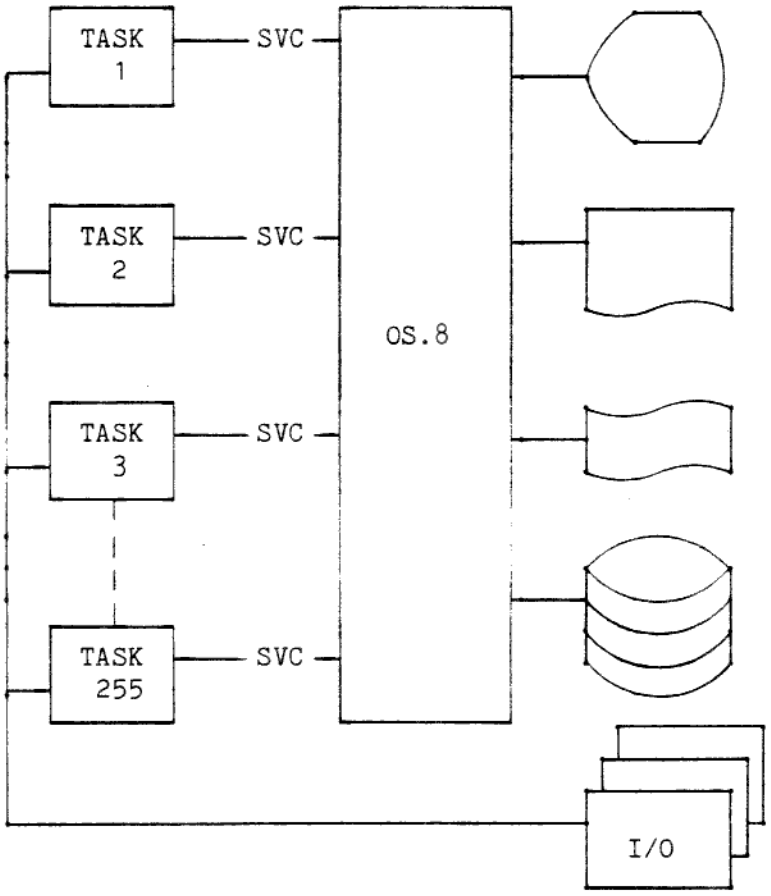

Figure: Principle System

CHAPTER 2

SYSTEM OPERATION

2.1 START-UP

The Operating System is normally loaded from a disc device via the System Bootstrap Loader, which loads a program sequence from the disc. This sequence then loads the Operating System.

To load, proceed as follows:

1. Mount an initiated disc containing the Operating System.
2. Reset the processor.

Possible error codes:

0. No OS image found.
1. All drives off line.
2. OS image larger than avalible memory.
3. Read error on OS image file.
4. Checksum error on OS image file.
5. OS image file not a contiguous file.
6. OS image must be located from zero and upward.

When the loading is complete, the identity is written on the terminal device as follows:

```
DOS.8MTM Rx.yz
```

Where x.yz is the release and update number, and the system is now ready to accept commands.

2.2 SHUT-DOWN

In a disc based system it is necessary that the system is shutted down in an orderly fashion, to assure the integrity of the discs in use. Before shutting down the system, the operator should cancel and delete all tasks, and close all opened discs.

CHAPTER 3

MULTI TERMINAL MANAGEMENT (MTM)

3.1 INTRODUCTION

The Multi Terminal Management (MTM) provides a multi-access environment for up to 8 concurrent users of suitable equipped DataBoard-4680 computer systems. The number of users is defined when configurating the MTM System. Each user operates at an interactive terminal, such as any TTY-compatible CRT, and has access to all the system resources and services needed to perform program development or real-time control. The MTM will load and run tasks with the priority given to a task at task establish time. Any program developed under the MTM will run unaltered in a suitable priviliged environment under OS.8.

The services available to a terminal user includes both a powerful Command Set and different Languages, as well as a number of Utility Programs. The command repertoire available to the MTM user gives immediate and straight forward control over the computing facilities available to the user.

The MTM System, in common with all multi-access time sharing systems, requires a certain amount of human operational involvement while it is in use. The actual amount will vary from installation to installation, and may be variable within an installation due to fluctuations in work load.

A valuable aspect of the MTM is its ability to support remote users. The MTM uses the standard input/output facilities of OS.8, which includes the use of remote terminals. Any or all of the MTM users can be remote from the computer installation, and dial-up lines can be supported.

3.2 ENVIRONMENT

Each terminal user is identified by a number <n>, where <n> is in the range 0 to 7, which is appended to the users task names.

The terminal is controlled by a task named Multi Terminal Console Monitor (MTCM). This task is responsible for the activation of the terminal assistent task when a terminal device becomes active.

When a user enters the MTM System, an assistent task is started by the MTCM. This task, which is named Command-handler (COMn), handles all the command and program execution for the user. The signon from the COMn task is:

```
DOS.8MTM Rx.yz User n entered YYYY-MM-DD HH.MM.SS
```

Where <x> is the revision level, and <yz> is the update level of the MTM System. The user is given the number <n>, and entered the MTM System at specified date and time.

Some of the MTM-commands will force a task to be loaded for command execution. The name of this task is UTLn (utility task user n).

When a program execution is requested, the program will be loaded and started as the primary task, and given the name User Program (USPn). The user may then interact with the task during its execution, or perform some command handling. This task can futher be referenced by some commands without specifying the task identity. (Ref CANCEL, CONTINUE etc).

3.3 TERMINAL USE

The MTM System is controlled by the user through a terminal device. This device can be any TTY-compatible device. It has a special relationship to the system in that the MTM System receives command input from the terminal and writes messages to it. Tasks may log messages to the user terminal without reference to its device name.

The name of the logical terminal device is always CON for every user, and may be assigned to a task for ordinary I/O operations, just as any other device.

3.4 PROMPTS

When the terminal operator is expected to enter data at the terminal, a prompt is output. This prompt takes one of the following forms:

-	Command Request (From COMn)
no prompt	Data Request

The command request prompt is output whenever the system is ready to accept another command.

The data request prompt is output whenever a task is attempting to perform a read request to the terminal device. This request should be satisfied as soon as practical, since messages are held in abeyance until the data request is satisfied.

3.5 CONTROL CHARACTERS

The control character conventions in effect for terminal devices are described below:

Deleting a Line
Depress simultaneously the CTRL-key and the X-key.

Deleting a Character
Depress simultaneously the CTRL-key and the H-key.

Ending an Input Line

If the input line is complete and ready to be processed, depress the carriage return key.

End-Of-File Function

To generate End-Of-File, simultaneously depress the CTRL-key and either the back-slash or Ö-key.

Break Function

If the data request prompt appears and the user wishes to communicate with the Command System rather than a task, the CTRL-key and the A-key should be depressed simultaneously. The system responds with the command request prompt, and is ready to accept a command. To return to the data request, just hit the Return-key.

If an input or output to the terminal is in progress, use of the break function will interrupt the process. For example, if the EXAMINE command has been entered and the output is in progress, the break function halts the output in progress. The system is then ready to accept a command.

Abort Function

Some commands and programs recognize an abort function, which aborts the terminal transfer and cancels the task execution. The abort function is generated by simultaneously depressing the CTRL-key and the C-key.

3.6 COMMAND HANDLING

The command is the basic unit of conversation between a terminal user and the MTM System. A command directs the MTM System to take a specific action. In general, a single command results in a single action being taken by the MTM System.

A command consists of a mnemonic which normally decribes the action the user wishes to take place, and arguments which provide the details necessary to perform the action.

Commands are accepted one line at a time. A command may not spread over two or more lines. A command line is terminated by a carriage return.

3.7 UNKNOWN COMMANDS

If an unknown command is entered, the MTM System tries to load a program with the same name. If found, it will be started as a primary task, and the rest of the command line will be transferred as parameters to that task.

3.8 ERROR RESPONSE

The MTM system responds to a command error by typing out a message to the user indicating the type of error. In response to the error

message the user must retype the entire command, correcting the error as necessary. The error messages are:

LOAD ERROR
Command or program not found. Memory space not enough, or checksum error on a command or program file.

SIZE ERROR
Additional memory size improperly specified.

SEQ ERR
This message is given if the particular command cannot be accepted due to the state of the system. This occurs either when a command is executing and another command is entered, or when a primary task is executing and execution of a new program is requested.

FD ERR
Syntax errror in a File Descriptor, or the type of the file is missing.

DEV ERR
Device not in system or not accessable.

ID ERR
Syntax error in a task name, or the task is not found.

PAR ERR
Parameter error, invalid or missing parameter.

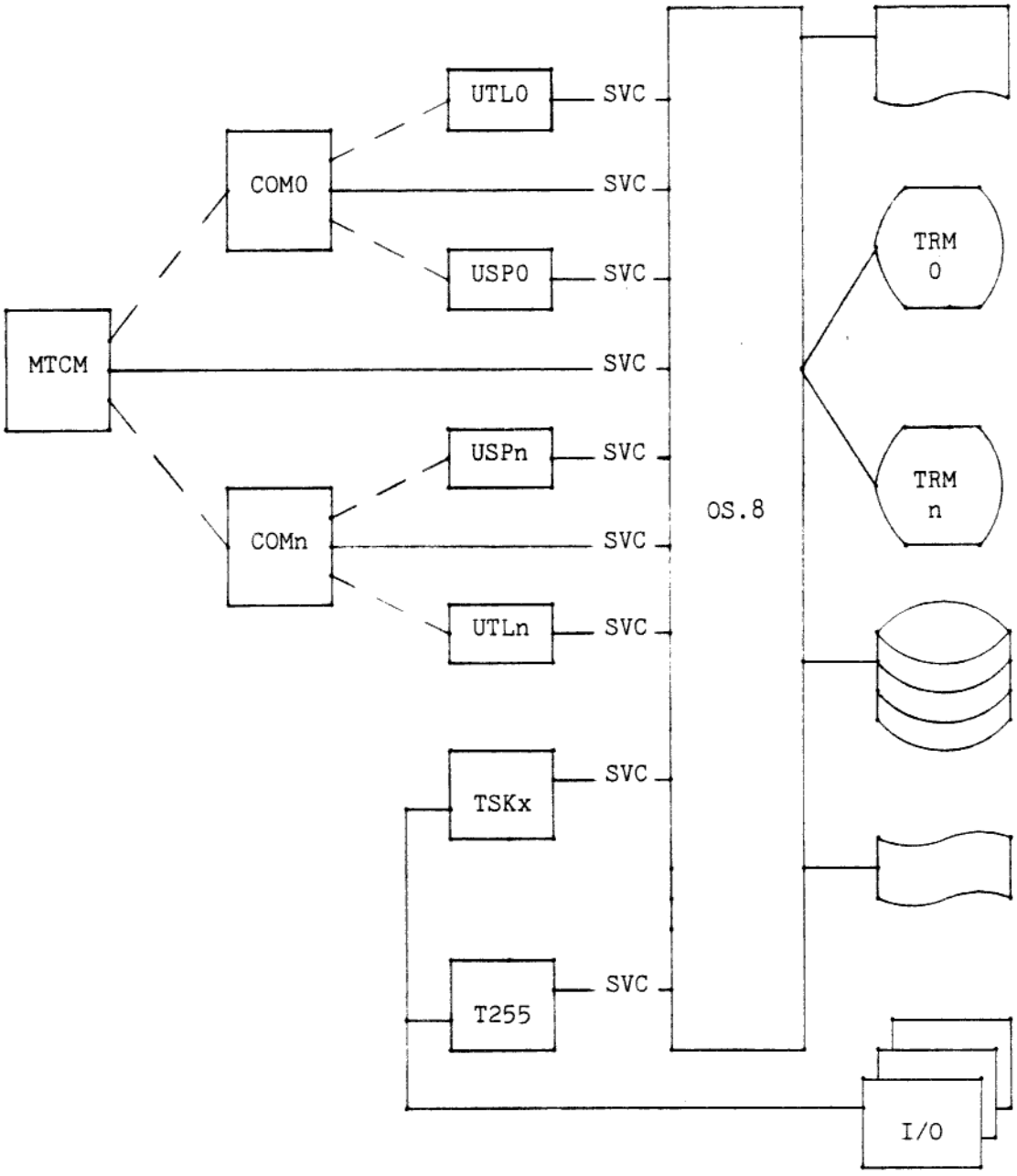

Figure: Principle MTM System

CHAPTER 4

COMMAND SYNTAX

4.1 GENERAL FORM

The general command structure, that is used in this manual, is as follows:

MNEMONIC(,(SWITCHES)(,ADDMEM)) ((PARAMETER1),(PARAMETER2),...)

Some commands have arguments that may or may not be specified. These arguments are enclosed in brackets in this manual. If the optional arguments are not furnished by the user, default values are supplied by the system. These default values are defined in the description of each of the command.

The following notations are used in this manual when describing the commands:

-capital letters indicates the minimum abbrevation for a mnemonic.

-punctuation marks and equal signs = must be entered exactly as shown.

-commas , separates arguments and substitute omitted positional arguments.

-brackets () indicate optional arguments.

-comma inside brackets (,) must be entered if the optional argument is chosen.

-slash / indicate alternative arguments, i.e, any one of the indicated arguments may be chosen.

-an elipsis ... indicates an optional repetition of the preceding argument.

4.2 MNEMONICS

Mnemonics are shown in this manual in a mix of Uppercase and Lowercase letters. A mnemonic may be entered into the system in its entirely or in an abbreviated form. The minimum abbreviations are indicated in this manual by Uppercase letters. Any number of characters between the minimum and the entire command may be entered.

For example:

The command TASK is given in the manual as TAsk. The following forms of this command may be entered:

```
TA
TAS
TASK
```

Illegal forms of this command are:

```
T      too short.
TAK    misspelled.
TASKS  too long.
```

4.3 SWITCHES

The switches follows the mnemonic immediately, and is separated from the mnemonic by a comma. The switches provides the user with ability to specify certain options in the form of unsequenced alphabetic characters related to the particular mnemonic. These switches are normally passed to the CPU registers of the task. Refer to OS.8 Program Reference Manual, SVC6, for futher information.

4.3.1 ADDITIONAL MEMORY

The <addmem> specifies the amount of extra memory, in bytes, to be added to a program. This memory expands the working area, which normally increases the execution speed of a program. The description of the commands and programs, specifies if any additional memory will be used or not.

4.4 PARAMETERS

The parameters are separated from the mnemonic and the switches by one ore more spaces. These parameters are transferred to the CPU stack of the task.

The following examples illustrate the use of the bracket:

```
COMmand(,ssss)  aaaa(, (bbbb) (,cccc) )
```

In this example, the operand aaaa is not optional, while the operands bbbb and cccc are optional. Note that the comma preceding operand cccc is also optional, but that the comma preceding operand bbbb is required, if operand cccc is specified.

```
ORder  xxxx(,yyyy (,zzzz) )
```

In this example, operand zzzz must not be entered without operand yyyy. This is shown by the nested brackets. Legal forms of this command are:

```
OR     xxxx
OR     xxxx,yyyy
OR     xxxx,yyyy,zzzz
```

Whereas in the previous example, legal forms are:

```
COM    aaaa
COM    aaaa,bbbb
COM    aaaa,bbbb,cccc
COM    aaaa,,cccc
```

4.4.1 NUMBERS

Numerical arguments are generally decimal rather than hexadecimal or octal numbers. One major exception is addresses which are expressed in chosen base.

Most numerical arguments are of integer form and thus, have no decimal point. Leading zeros may be omitted in numerical arguments, whether decimal, octal or hexadecimal.

Octal numbers are expressed in a split form. This means that the octal value is divided into two parts, separated from each other by a colon. In the following example, all values are equal:

```
7214    Decimal
1C2E    Hexadecimal
34:56   Split Octal
```

4.4.2 TASK IDENTIFIERS

A task identifier (indicated as TID in this manual) consists of one to four alphanumeric characters, the first of which must be alphabetic.

Valid task identifiers are:

```
TSK3
MAX
X
T9X5
```

Invalid task identifiers are:

```
3TSK  first character not alphabetic.
T4.2  nonalphanumeric character.
```

4.4.3 FILE DESCRIPTORS

A file is a program or a collection of data stored on a direct access storage device. Files stays in the system permanently unless they are explicity removed. Files are identified in the system by a File Descriptor. The File Descriptor contains the name of the volume on which the file reside, the file name or the element directory name, the element name, and the type of the file.

File Descriptors may refer to devices as well as to direct-access files, in which case, the VOLN field is the four-character device mnemonic. A FILENAME and ELEMENT, if entered, are ignored. The colon following VOLN must always be entered in this case.

File Descriptors (abbreviated FD in this manual) are composed of four fields:

VOLN:FILENAME.ELEMENT/TYPE

VOLN is the name of the volume on which the file resides, if FD refers to a file. It may be from one to four characters. The first character must be alphabetic and the remaining alphanumeric. The volume need not to be specified, and the default volume is then specified by the SYSTEM volume.

FILENAME is the name of the file. It may be from one to twelve characters, the first alphabetic and the remaining alphanumeric.

ELEMENT is the name of an element in a element directory. It may be from one to twelve alphanumeric characters.

TYPE describes the type of the data within a file. Refer to chapter titled FILE TYPES.

Example of legal File Descriptors are:

```
PACK:MAIN.LOOP
MAIN.LOOP       the same, if PACK is the system volume.
CARD:           name of a device
```

4.4.4 FILE TYPES

With each file there is a type specification that describes, for the system and the user, what kind of data there is in the file. The type is specified by one or two either alphabetic characters, or hexadecimal digits, where the leftmost has the highest priority. The type of the file is normally implied by the programs, and does not need to be specified. If the file type is not implied by the program, it must be specified!

The first character or digit specifies the type of data in the file:

```
0x/U  Und, undefined data, which verifies to any other type.
1x/A  Asc, ASCII data readable without any special handling.
2x/L  Lst, list file, ASCII data together with position information.
3x/O  Obj, object code, readable by the Task Establisher. Cannot be
           loaded and executed.
4x/B  Bin, binary data, which is unspecified.
5x/T  Tsk, task file, either relocatable or absolute. Can be loaded
           and executed.
6x/I  Ism, ISAM index file.
7x         Reserved.
 :          -   "   -
Fx         Directories.
```

The second character or digit, if entered, is mostly used for information purpose and defines a set of languages and directory types:

x0/U	Und, undefined data, verifies to any other subtype.
x1/A	Asm, ASSEMBLER source code.
x2/B	Bas, BASIC source code, or data produced by BASIC.
x3/C	Cob, COBOL source code, or data produced by COBOL.
x4/F	Ftn, FORTRAN source code, or data produced by FORTRAN.
x5/P	Pas, PASCAL source code, or data produced by PASCAL.
x6	Reserved.
:	- " -
FD/D	Efd, Element File Directory.
FE	Reserved.
FF/D	Mfd, Master File Directory.

4.4.5 WILD-CARD

Some commands may handle generalized File Descriptors, which are used to specify a searching key, when scanning for one or several files.

Special formats:

An asterisk * is a position dependent replacement for any character, and may be placed anywhere except after a dash.

A dash - is a position independent replacement for any character including space, and must be the last character in a certain field.

Example of valid forms are:

ABC** Any file that starts with ABC and has five characters in its name.

BC- Any file that starts with any charcter, followed by BC, and is at least four characters long.

CHAPTER 5

MTM COMMANDS

5.1 INTRODUCTION

This chapter describes the commands available to the terminal user. These commands are grouped according to their function. Most of the commands are performed by Supervisor Calls (SVC), which makes them available to the programmer. More detailed information will be found in OS.8 Program Reference Manual.

5.2 GENERAL SYSTEM COMMANDS

The following commands pertain to the system as a whole. As such, they have a global effect on the system or display global system information:

SLICE	Handles the time-sharing function.
TIME	Affects the time of day.
VOLUME	Specifies the system volume.

5.2.1 SLICE Command

Two types of scheduling algoritms are available. Tasks may be scheduled in strict priority order or time-sliced within priority. In the former case, if two tasks of equal priority are started, a task remains active until it relinquishes control of the processor. Care should be taken in assigning priorities so that tasks which do not frequently relinquish control of the processor do not inadvertently lock out other tasks. A task may relinquish control in one of the following ways:

- It is paused or cancelled by the console operator or another task.
- A higher priority task becomes ready because of some external event.
- It executes an SVC that places it it Wait, Paused or Dormant state.

Rather than scheduling on a strict priority basis, tasks may be time-sliced within priority. This option allows the user to ensure that tasks of equal priority receive equal shares of processor time.

When a task becomes ready, it is queued on a round-robin basis behind all ready tasks of equal priority.

The SLICE command is used to invoke the time-slice scheduling option. Its format is:

```
SLice (n)
```

where: <n> is a decimal number specifying the time slice in milliseconds.

If <n> is omitted, the current slice value will be displayd. If <n> is 0, time-slice scheduling is disabled, otherwise, <n> represents the the maximum time, in milliseconds, any task can remain active if another task of equal priority is ready.

The time-slice option is initiated at the system generation time.

5.2.2 TIME Command

The TIME command should be entered when the system is loaded. It may be entered at any other time that the system clock is incorrect. The day, month and year are automatically updated by the system, even during leap years. The format of this command is:

```
TIme (YYYY-MM-DD, HH.MM.SS)

  where:
    YYYY=year
      MM=month
      DD=day
      HH=hours, 24-hour clock.
      MM=minutes
      SS=seconds
```

If <YYYY-MM-DD, HH.MM.SS> is omitted, the current time will be displayed.

If a TIME command is entered while there are uncompleted time intervals (see SVC 3), the tasks which initiated the incomplete intervals are affected in the following way:

Seconds and milliseconds from now. Elapsed time intervals are unaffected by a change in the time. Time-of-day requests will not elapse at the right time.

5.2.3 VOLUME Command

The VOLUME command is used to set or change the name of the system volume. Alternatively, it is used to interrogate the system for the current name associated with the system volume. The format of this command is:

```
Volume (voln)
```

where <voln> is the system volume identifier.

No test is made to ensure that the volume is actually on-line at the time the command is entered. If <voln> is not specified, the name of the currently default system volume is output to the console.

The following is an example of a VOLUME command:

```
-V
 SYSTEM
 DOS8
```

5.3 UTILITY COMMANDS

This group of commands is useful in debugging and memory manipulation.

BIAS	Sets a base address.
EXAMINE	Displays the main memory.
MODIFY	Changes the main memory.
RADIX	Sets the base.

5.3.1 BIAS Command

The BIAS command is used to set a base address for the EXAMINE and MODIFY commands. Its format is:

```
BIas address
```

The operand <address> is a bias to be added to the address given in any subsequent EXAMINE or MODIFY command.

A BIAS command overrides all previous BIASes.

To find out which the current BIAS is, enter EXAMINE 0.

The operator should enter a BIAS command if the current value is unknown.

5.3.2 EXAMINE Command

The EXAMINE command is used to examine the content of memory. The formats is:

EXamine address(,n)

The EXAMINE command causes the contents of the memory location specified by <address> (as modified by any previous BIAS command) to be displayed. The operand <n> specifies the number of bytes to be displayed, and is given in the same radix as the <address>. If <n> is omitted, two bytes are displayed.

5.3.3 MODIFY Command

The MODIFY command is used to change the contents of memory. Its format is:

```
MOdify address,data(,data...)
```

This command causes the contents of the byte location specified by <address> (modified by any previous BIAS command) to be replaced with <data>. Each <data> field represents a byte to be written into the memory starting at the location specified by <address>.

5.3.4 RADIX Command

This command is used to select the notaion of the numeric values used in all the utility commands. The format of this command is:

RAdix 8/16

If radix 8 is selected, then all address values will be represented in split-octal.

5.4 TASK RELATED COMMANDS

The following commands are related to particular tasks. If the TID parameter is optional, the primary task USPn is assumed when omitted.

CANCEL	Abort the execution of a task.
CONTINUE	Continue the execution of a paused task.
LOAD	Load a program into the main memory.
OPTIONS	Change the options of a task.
PAUSE	Pause an executing task.
PRIORITY	Change the priority of a task.
RUN	Load and start a program
START	Start a task.
TASK	Display the present tasks and their characteristics.

5.4.1 CANCEL Command

The CANCEL command terminates a task as if it had executed an SVC-6 with cancel as function code and 0 as cancel code. The format of this command is:

CAncel (tid)

If the task is non-resident, it is removed from the system memory. All outstanding I/O must terminate and then the task's Logical Units (LU's) will be closed. If the task is resident, it is not removed from the system memory.

This command may be entered even when the specified task is Dormant. It has no effect on a resident task that has already gone to End-of-Task (EOT), unless preceded by an OPTION,N command. If preceded by an OPTION,N command, it can be used to remove a task from the system memory.

5.4.2 CONTINUE Command

CONTINUE causes a task which has executed a Pause SVC, or has been PAUSed by the operator to resume operation. The format of this command is:

```
COntinue (tid)
```

5.4.3 LOAD Command

A task must be prepared by processing the component programs and subroutines with ESTAB program. Once established, the task can be loaded. A task is loded into the first memory segment large enough to accommodate it. The format of this command is:

```
LOad fd(, (tid)(,size) )
```

The <fd> is the file descriptor of the file containing the established task, and the <tid> is the task identifier that the task is to be known by once it is loaded. If <tid> is omitted, it defaults to the first four characters in <fd>. The <size> is the amount of area in the tasks impure memory segment. It is specified as a decimal number in bytes, and if omitted, default is 0.

If load is performed from a device, the logical record length must be 256.

5.4.4 OPTION Command

The OPTION command, available from revision 3.00, is used to change certain options of the specified task. The format of this command is:

OPTion,opt(opt...) taskid

where <opt> may be any of the following options:

R = resident, the task is memory resident.

N = nonresident, the task is to be removed at End-Of-Task.

A = abortable, the task is cancellable from other tasks.

P = protected, the task is not cancellable from other tasks.

5.4.5 PAUSE Command

The PAUSE command causes the specified task to pause. The format of this command is:

PAuse (tid)

Any I/O-Proceed, on going at the time the task is paused, is allowed to go to completion. If the task is in any Wait state at the time the PAUSE command is entered, all external wait conditions must have been satisfied before the PAUSE becomes effective. This command is rejected if the task is dormant or paused at the time it is entered.

5.4.6 PRIORITY Command

The PRIORITY command, available from revision 3.00, is used to modify the priority of the specified task. Its format is:

```
PRIority tid,n
```

where <n> is a decimal number from 10 to 255 inclusive.

5.4.7 RUN command.

The RUN command is used to load and start a program. The format of this command is:

RUn(,switches) fd(,parameters)

The program is loaded from <fd>, and the task is given the first four characters in <fd> as its name. Then the task is stared, and the <switches> and the <parameters> are passed to the task.

5.4.8 START Command

The START command is used to initiate task execution. The task specified is started only if it is dormant. The format of this command is:

```
STart(,switches) tid(,parameters)
```

The <tid> is the name of the task, and may consist of from one to four alphanumeric characters.

The optional fields <switches> and <parameters> contains arguments that are to be passed to the task.

5.4.9 TASK Command

The TASK command causes a map of the present tasks to be output to the console or the <fd>. The format of this command is:

TAsk(,F) (fd)

The following is an example of a TASK command output:

```
-TA,F
TASK   NR  STAT  TYPE  PROGRAM      PRI  TCB-ADR  SIZE     ENTRY
MTCM    1  W     RN                 20  367:124  010:000  123:456
COM0    2  W                        30  364:324  011:222  234:321
UTL0    3                          128  235:164  037:765  236:124
```

The TASK field contains the symbolic name of the task, and the NR field is the system task number.

The STAT field contains the current status of the task, which can be:

DORM	dormant, not started.
C	cancel pending, on its way to terminate.
P	paused
S	suspended, ready to execute.
W	waiting, for an event.

The TYPE field indicates the type of the task:

E	E-task
N	non-abortable
P	pure code
R	resident

The PROGRAM field is reserved, and PRI is the currently priority assigned to the task.

If fully display is requested (switch <F>) the TCB-ADR field contains the address of the TCB for the task, the SIZE field holds the memory size allocated to the task, and the ENTRY field is the default start address of the task.

5.5 DEVICE AND FILE CONTROL COMMANDS

The following set of commands are used for device and file control.

ALLOCATE	Create a direct-access file.
CLOSE	Take a device off-line.
DELETE	Remove a direct-access file.
DEVICES	Display the present devices.
LIB	Display the files on a volume.
OPEN	Take a device on-line.
POSIT	Position a device.
RENAME	Change the name of a direct-access file.
SPACE	Examine the space available on a volume.

5.5.1 ALLOCATE Command

The ALLOCATE command is used to create a direct-access file. The following formats exist for this command:

(a) ALlocate(,I) fd(, (lrecl) (, (size)(/blk)))

(b) ALlocate,C fd,size

The operand <fd> identifies the file to be allocated, where the file type must be specified. Format (a) is used to allocate an Indexed file, and format (b) is used to allocate a Contiguous file.

If INDEX is chosen, the next operand, <lrecl>, is optional and specifies the logical record length. It cannot exceed 65535 bytes. Its default is 0 bytes (variable record length). The file size operand, <size>, is optional and specifies the total allocation size in sectors. The <blk> operand specifies the physical block size in sectors.

If CONTIGUOUS is chosen, the file size operand, <size>, is required and specifies the total allocation size in 256-byte sectors. This may be any value up to 65535*CLUSIZE, or the number of contiguous free sectors existing on the specified volume at the time the command is entered. The <size> is specified as a decimal number. Refer to the DISKINIT program description for futher details about file parameters.

Examples of the ALLOCATE command:

```
AL THISFILE/B,126
```

This example allocates, on the system volume, an Indexed file named THISFILE/Bin with a logical record length of 126 bytes and default preallocated.

```
AL,C NEW:BIGFILE/A,5000
```

This example allocates, on the volume NEW, a Contiguous file named BIGFILE/Asc whose total length is 5000 sectors.

5.5.2 CLOSE Command

The CLOSE command is used to take a device off-line. The format of this command is:

```
CLose fd
```

The mnemonic name of the device is specified by <fd>. The CLOSE command is rejected if it is directed to a device which is currently assigned.

If the device being CLOSEd is a direct-access device, the <fd> used in the command is not the volume identifier, but the actual device mnemonic. For example, to CLOSE a disc named FPY1: which currently contains a volume named DOS8:, the operator enters:

```
CLOSE FPY1:
```

This action removes the volume DOS8: from the system. The disc may now be removed or changed. If a direct-access device is dismounted without being CLOSEd, it may only be OPENed on-line in the write protect mode. (Valid from 3.00). This will ensure that the disk can not be assigned for writing untill its structure has been checked out by the DISKCHECK program.

5.5.3 DELETE Command

The DELETE command is used to delete a direct-access file. Its format is:

```
DELete fd(,fd...)
```

The <fd> identifies the file to be deleted. Type must be specified. To be deleted, the file must not be currently assigned to any LU of any task.

5.5.4 DEVICES Command

The DEVICES command allows the operator to determine the name, number, status, type, volume name, current request, channel number, interrupt level and the address of the Device Control Block (DCB) of all devices in the system. The format of this command is:

```
DEVices (fd)
```

The optional operand <fd> specifies the device or file to which the display is routed. If omitted, the display goes to the terminal device.

The following is an example of a DEVICE command output:

```
-DEV
MNEM   NR  STAT  TYPE  VOLN  DCB-ADR   REQ   SVC-BLK   CS  IL
NULL    0
RE      1                   275:310   USP0  123:345   77   5
PR      6                   275:034                 74   5
PU      7                   274:306                 76   5
FPY0    8        DIR   DOS8 267:030                 40   6
FPY1    9  OFFL              266:266                 40   6
TRM0   64                   265:342                 75   4
CON    65                   275:162
```

The MNEM field contains the symbolic name of the device, and the NR field contains the system device number.

The STAT field gives the information about the status on the device, which can be:

```
OFFL  off-line
PROT  write protected
      on-line
```

The TYPE field indicates:

```
DIR   directory oriented device
TASK  task device
      physical device
```

The VOLN field contains either the volume name of a directory oriented device, or the name of the Symbiont task that owns the device.

The CS field contains the Card Select code, the IL field contains the Interrupt Level and the DCB-ADR field the address of the DCB for the devices. The REQ field contains the name of task currently accessing the device, and the SVC-BLK field holds the address to the involved parameter block.

5.5.5 LIB Command

The LIB command is used to display the content of a volume or an element file. The format of this command is:

```
Lib(,F)  ((VOLN:)(ELEMENTDIR),(FILESPEC),(LIST))
```

The switch <F> is used to display all information about each file, such as length, creation date etc.

The VOLN is the name of the volume, from which the display will be done. If omitted, the system volume will be used.

The ELEMENTDIR is the name of the Element File Directory, that should be examined.

The FILESPEC is the name of a unique file, or a wild-card specification of a group of file names, to be displayed. If omitted, all files will be displayed.

The LIST is the name of the device or the file to whom the listing should be directed. If omitted, the terminal device is assumed.

5.5.6 OPEN Command

The OPEN command is used to bring on-line a device that was previously off-line. The format of this command is:

```
OPEn(,(N)(P)) fd
```

The mnemonic name of the device is specified by <fd>. After opening a direct-access device, the volume name associated with it is output to the console device.

If the switch <P> is specified, the device is opened as write protected. If the device is hardware write protected, it will be automaticly opened protected.

The switch <N> is used to open a direct-access device non-filestructured. That means that no directory is present and that no volume name will be established at open.

While a device is off-line, it cannot be assigned to any task.

If the device being OPENed is a direct-access device, the <fd> used in the command is not the volume identifier, but the actual device mnemonic. For example, to make the new volume known to the system, the operator enters:

```
OPEN FPY1:
```

5.5.7 POSITION Commands

This set of commands allows the operator to manipulate with magnetic tapes and cassettes from the console. The format of this command is:

```
POSit fd,cmd(,n)
```

The command <cmd> will be executed <n> times on <fd>, and if <n> is omitted, the command is executed one time.

The command <cmd> may be one of the following:

BRecord, backspace one record.

BFile, backspace to filemark

FRecrod, forward-space one record

FFile, forward-space to filemark

REWind, rewind

RW, rewind

WFile, write filemark

5.5.8 RENAME command.

The RENAME command is used to change the name of an unassigned direct-access file. Its format is:

```
REName oldfd,newfd
```

If an element file is to be renamed, the directory name does not need to be specified in <newfd>.

Note that RENAME FIL/A,FILE/AB is not possible since the main file type does not change.

This command cannot be use to rename a direct-access volume. The DISKINIT program must be used for this.

5.5.9 SPACE command.

The SPACE command is used to examine the space available on a direct-access volume. Its format is:

```
SPace (voln)
```

The optional parameter <voln> is the name of the volume to be examined. If omitted, the system volume is assumed.

OS.8

UTILITY REFERENCE MANUAL (URM)

CONTENT

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

This manual is designed as a reference manual for the utility programs available. The system requirements and running procedures is described for each program.

Use of this manual requires that the reader is familiar with the features, functions and conventions of the DataBoard-4680 system from the user´s point of view as documented in:

- DataBoard-4680 OS.8 Terminal Reference Manual.

This publication is divided into four more chapter groups:

Chapter 3-7 describes the disc maintenance programs.

Chapter 8 describes some copy programs.

Chapter 9-15 holds a set of development tools.

Chapter 16-20 describes the language processors.

SYSTEM BOOTSTRAP LOADER (BOOTLOADER)

2.1 INSTALLATION

The System Bootstrap Loader, BOOTLOADER, is available in two versions, with or without an interactive part. It is normally delivered without the interactive part, which has to be ordered specially.

The BOOTLOADER is contained in two fusible link PROMs to be inserted at location 3B, 3C on a Control Board (1044). The Boot Enable switch located at the middle of the board should be in the close position.

2.2 OPERATION

When power are aserted or the Reset switch is pressed, the CPU enters the Bootstrap Mode. The first step is to locate the highest writable memory location, by scanning it downwards in 2KB steps. Then the stackpointer is initialized to this address.

The second step, depending on the position of the switch located at the Interval Clock Board (5018), is to enter, either the Automatic Disc scan Boot sequence, or, depending on the presens of a console interface, the Interactive or Character Boot Mode. The interactive part is optional, and has to be ordered specially.

If the switch is in the lower position the disc scan mode is entered and all possible disc devices are scanned in the following order:

1 Miniflexible disk, CS 55, Dev 0-1
2 Flexible disk, CS 40, Dev 0-3
3 10 MB Cartridge disk, CS 44, Dev 1,0,3,2
4 Flexible disk 4611, CS 45, Dev 0-3
5 67 MB Cartrideg disk, CS 46, Dev 0-3
6 Winchester disk, CS 47, Dev 0-3

If any device is found on line and Boot loading was successfull, control is transferred to location 0100H.

If the switch is in upper position and the interactive part is included, the presens of a console device is checked, and if found, the Interactive mode is entered. Else the Character device load mode is entered. This mode allows booting from a character device in either Binary or Hexadecimal format. Character input devices are:

1 Card Reader
2 SP1 Reader
3 Console UART input

These devices are scanned until one of them becomes ready (has data) and that device is selected for the further reading. If the load

appears to be OK, transfer is done to the specified address, otherwise the program halts.

In the Interactive mode there are a wide range of possibilities to affect the booting. Some small things should however be kept in mind:

- If the message "ERR" is given, the booting should be restarted from the very beginning.
- Each keyword and parameter is followed by a colon, which acts as both delimiter and terminator.
- Leading zeros need not necessary be typed in, except when retyping a number, e.g. the three last didgits are always valid. All numbers are in octal.
- The dump printout is arranged for printers and if directed to the console, the last part of the line will be lost if "AUTOLF" is not supported.
- To abort a not completely typed command, the CTRL + C keys should be strucked.

2.3 COMMANDS

If the interactive part is included, and the switch located on the Interval Clock Board (5018) is in the upper position, the Interactive mode is entered, provided that there is a UART board (4017) with the Selection Code 75Q installed. The following commands are avaible:

READ:
When given, Cardreader, SP1-Reader and Console are scanned in order until any of the devices becomes ready. From that only, that device is used for data input.

DISK:type:cs:dev:
This command is used to transfer the boot loader portion from the specified disk device into memory. The parameter list are as follows:

type:
The type of disc-interface to be used:
H:
Disk devices connected with the common disk interface board (4109).
F:
Single density, single sided flexible disks connected with interface board (4034).
M:
Single density, single sided minflexible disks connected with interface board (4076).

cs:
Select Code for desired interface board, refer to summary for standard device selection.

dev:
Logical unit selection.

START:
When given, control is transferred to the address obtained from the previously used load routine. For disk devices, the transfer address is always 0100H.

BEGIN:adrh:adrl:
Similar to the START command, except that the transfer address is given as argument. Address format is always expressed in split octal.

DUMP:radix:cs:SAH:SAL:LH:LH:
This command is used to dump a selected portion of the memory in the selected radix. Radix can be either H: for hexadecimal or O: for octal dump. The list device selectcode is <cs> and can be either a SP1 (4015) or UART (4017) board. SAH, SAL is the start address expressed in octal, and LH, LL is the length also expressed in octal.

2.4 SUMMARY

Command Summary

READ:	Load from character device.
START:	Transfer to obtained address.
BEGIN:adrh:adrl:	Transfer to specified address.
DISK:type:cs:dev:	Read boot from specified disk.
DUMP:radix:cs:SAH:SAL:LH:LL:	Dump memory area.

Table Over Device Selection

Description	Type	CS	Dev	Remark
Mini flexible disk	M	55Q	0-1	4076 5"
Flexible disk	F	40Q	0-1	4034 8"
Flexible disk	H	45Q	0-7	4611
10 MB Cartridge disk	H	44Q	0-7	4610-6
67 MB Cartridge disk	H	46Q	0-3	4610-7
Winchester disk	H	47Q		4612
Interactive Console		75Q		4017
SP1-Reader		77Q		4016
Card Reader		70Q		4037
System Printer		74Q		4015/4017

DISK FORMATTER (DISKFORM)

3.1 INTRODUCTION

The Disk Formatter (DISKFORM) generates the magnetic structure on the diskpack, such as the pre-amble with the address mark, a data section with dummy data, the post-amble containing the checksum and the inter-record gap.

3.2 SYSTEM REQUIREMENTS

The program requires:

-10 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

3.3 STARTING THE DISC FORMATTER

To initialize a disc pack, first set the disc off-line using the CLOSE command. Then start the program by the command:

```
DISKFORM
```

When started, the program displays a menu of available commands, which are given in an interactive mode.

3.4 REQUIRED COMMANDS

The following keywords are required, and must be specified in the following order:

DEvtype=t
: The type of device, which is presented in a menu by the program.

DRive=fd
: Specifies the device, where <fd> is the name of the device..

3.5 OPTIONAL COMMAND

The following commands are optional:

FIll=n
: Where <n> is a decimal value to be filled into all sectors. Default is is 229 (E5H).

INterval s(-e)
: This command is used to specify the track <s> to be formatted, or the track interval starting at <s> and ending at <e>. Default is all the tracks.

3.6 INTERACTIVE COMMANDS

In the interactive mode there are four more commands:

HElp
: Displays the available commands.

PArameters
: Will display the actual setting of formatting parameters. This command will not affect any parameters.

STart
: Will start the formatting.

ENd
: Terminates the program execution.

3.7 MESSAGES

The program may output the following messages:

DISK FORMATTER Rx.yz
: Signon by the program, where the revision level is <x>, and the update level is <yz>.

COMMAND ERROR
: When an unknown command is entrered.

ASSIGN ERROR
: Failed to assign the device specified in the DRIVE command.

NOT IMPLEMENTED IN THE VERSION
: The device type specified in the DEVTYPE command is not supported.

INTERVAL OUT OF DISK
: The interval specified within the INTERVAL command is outside the disk.

UNDEFINED DRIVE
: The required command DRIVE has not been entered.

UNDEFINED DEVICE
: The required command DEVTYPE has not been entered.

DISK NOT READY
Time-out on the disk drive.

I/O ERROR <s>
When an error is detected during disc read or disc write. The return status is <s>.

END OF TASK <s>
The program terminates, where <s> may contain the SVC error status.

3.8 EXAMPLE

This is an example how to format a disk pack:

-DISKFORM	Load and start the program.
Disk Formatter Rx.yz	Signon by the program, then the command menu is displayed on the terminal.
DEV=CART	Device type is 10 MB cartridge.
DR=DP0:	Drive specification.
ST	Start formatting all the tracks.
END	Terminate the task.
-12.34.56 End-of-task 0	The program terminates.

DISK INITIALIZER (DISKINIT)

4.1 INTRODUCTION

The Disk Initializer, DISKINIT, initializes a disc pack for use with the Operating System. Initialization includes placing the Volume Name and a pointer to the Bit Map and Master File Directory on the Volume Descriptor, which is on the first sector of the disc.

The Volume Name consists of one to four characters, the first of which must be alphabetic. This name identifies the disc to the system. The DISKINIT allows a disc pack to be named or renamed.

The Master File Directory describes all files on the pack. Filenames and starting sector address identify each file on the disc.

The Bit Map contains one bit for every cluster on the pack. If a bit is set, the cluster it represents is allocated. Files are allocated on free sectors in cluster quantities and these clusters are then marked as used in the Bit Map. The DISKINIT allows the user to clear the Directory and Bit Map in order to delete all files on the disc. DISKINIT also provides a facility to clearing a new disc pack.

4.2 SYSTEM REQUIREMENTS

The DISKINIT program requires:

-14 KB of memory above the operating system, including buffer for the Read Check operation.

-a console device.

-a currently supported disc device.

4.3 START OPTIONS

Six options are available with the DISKINIT program: CLEAR, NOREADCHECK, CLUSIZE, BLOCKSIZE, DEFAULT and DIRECTORY.

Whenever command CLEAR is specified, a read check operation checks the disc for bad sectors, unless the command NOREADCHECK is given. These sectors are marked as unavailable in the Bit Map. If the first sector on the disc fails the read check operation, the pack cannot be used and a message to that effect is printed. Because media degradation may occur at any time, there are instances where sectors which are not flagged at format time are flagged as bad sectors during initialization. In such instances, it is recommended that the disc pack be backed-up, re-formatted and re-initialized. The data can then be restored.

The command NOREADCHECK could be given to speed up initialization, and the disc will not be checked for bad sectors. This command should not be used if the disc is not known to be "top shape", and is primary used for test purposes.

The CLUSIZE command is used to specify the smallest allocatable element on the disc, and is given in sectors. The value must be a power of 2. E.g. 1, 2, 4, 8, 16 etc. Max value is 128.

The BLOCKSIZE command is used to specify the default blocksize in sectors, at SVC7 allocate, (refer to SVC7 FILE HANDLING in OS.8 Program Reference Manual). The value given is rounded off to a multiple of the CLUSIZE. The value may range from 1 to 255.

The DEFAULT command is used to specify the number of blocks specified in BLOCKSIZE to be allocated as default, if not given in the SVC7 Block. The value given may range from 1 to 65535.

The DIRECTORY command may be used to locate the Master File Directory (MFD) on an other than default location on the disc. A second parameter is used to specify the directory size in blocks of 16 sectors. Currently only 1 as size is supported.

4.4 STARTING THE DISC INITIALIZER

To initialize a disc pack, first put the disc off-line using the CLOSE command. Then open the disc NON-FILESTRUCTURED by the command OPEN,N fd.

All information required by the DISKINIT is specified within the DISKINIT command or if no start parameters are given in Interactive Mode:

```
DISKINIT (DEV=t,DR=fd,V=xxxx(,....))
```

4.5 REQUIRED COMMANDS

The following keywords are required, and must be specified in the following order:

DEVtyp=t
: The type of device, which is presented in a menu by the program.

DRive=fd
: Where fd is the name of the disc device.

Volume=xxxx
: Where xxxx is the volume name to be given to the disc pack.

4.6 OPTIONAL COMMANDS

The following commands are optional, and may be entered in any order:

CLEar
: Specifies a clear disc and read check operation. When entered, all files are deleted from the disc. A read check of the entire disc is performed and bad sectors are flagged in the Bit Map. All flagged sectors are identified in a message by their decimal sector addresses on the disc.

Noreadcheck
: Specifies whether the disc should be checked for bad sectors or not. If given, no read check will be performed. Should be used with care.

Readcheck
: Similar to NOREADCHECK, but will turn readcheck on.

CLUsize=n
: Specifies clustersize in sectors where n must be a power of 2. This command will affect the size of the Bit Map.

Blocksize=n
: This parameter specifies the default block size in a file, and is given in sectors. Number of sectors is, if not a multiple of clustersize, rounded off at SVC7 allocate. <n> may be in the range from 1 to 255.

DEFault=n
: The parameter given will specify the number of blocks to be preallocated at allocation time. The value <n> is the number of blocks ranging from 1 to 65535.

DIRectory=lsa/n
: This command is used to change the default location/size of the Master File Directory. This command should be used when any of the system sectors is found to be bad. The starting sector should be chosen so it will not interfere with any bad starting sector. The size in sectors is calculated as (n*16+1+bitmapsize) divided by (CLUSIZE*CLUSIZE+CLUSIZE).

4.7 INTERACTIVE COMMANDS

In the interactive mode there are three more commands:

PARameters
: Will display the actual setting of initialization parameters. This command will not affect any parameters.

STart
: Will start the initialization.

ENd
: Terminates the program execution.

4.8 MESSAGES

The DISKINIT outputs the following messages:

DISKINIT Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

MORE THAN 65535 SECTORS FLAGGED, FORMAT DISC
Readchecks founds an extremly bad disc. The disc should be formatted prior to use.

nnn SECTORS FLAGGED
Informs the operator how many sectors that where flagged in the readcheck operation.

NO SECTORS FLAGGED
Informs the operator that no sectors on the disc was found bad.

DIRECTORY WRITE ERROR
Is given when the program fails to write on the disc. Write protect switch should be checked.

SYSTEM SECTOR FLAGGED IN READCHECK
Is given when the program sould be re-runed, and the DIRECTORY command should be used to locate the directory on another starting position.

READCHECK ERROR! LOGICAL SECTOR NNN(10) FLAGGED OFF
Informs the operator that one sector is found to be bad and is marked busy in the Bit Map. NNN is the logical sector address on the disc.

DISKERR FC=ffQ RETURN STATUS ssQ
Is given when an unexpected error occurs at disc read or write. FF is the function code in octal; SS is the return status in octal.

CMD-ERR, TYPE=tttt
Informs the operator that a command was not recognized or in bad format. TTTT is the error specification as:

DEV-TYPE, device type not found in the default table.
VOL-NAME, bad format on volume name.
TOO MANY ARGUMENTS, expected comma or end, found another parameter.
INVALID PARAMETER, bad format or can´t be accepted due to range.
INVALID FD, bad format or device not found.
UNKNOWN COMMAND, command not recognized.
MISSING NONDEFAULT PARAMETER, the START command was given and any of the commands DEVICE, DRIVE or VOLUMNE was not given.
CAT OUT OF DISC, the start position given in DIRECTORY command will not give sufficient space for the Bit Map sectors.
SEQUENCE, the non default commands are not given in proper order.

4.9 DEFAULT INITIALIZATION PARAMETERS

DEVICE TYPE	CLUSIZE	BLOCKSIZE	DEFAULT	DIRECTORY
FLOPPY 5" Normal	1	4	1	8/1
FLOPPY 5" Double	1	4	1	16/1
FLOPPY 5" Quad	1	4	1	32/1
FLOPPY 5" Oct	1	4	1	64/1
FLOPPY 8" SS, SD	1	1	16	130/1
FLOPPY 8" SS, DD	1	1	16	247/1
FLOPPY 8" DS, SD	1	1	16	260/1
FLOPPY 8" DS, DD	2	1	16	507/1
HARD DISK 10MB	2	4	8	9600/1
WINCHESTER 10MB	2	4	8	12800/1
WINCHESTER 20MB	2	4	8	12800/1

4.10 EXAMPLE

This is an example of an initialization of a discette.

-CLOSE FPY1:	Close the drive if it is opened. Then mount the disc to be initialized.
-OPEN,N FPY1:	Open the drive non-filestructured.
-DISKINIT	Load and start the diskinit program.
Diskinit Rx.yz	Signon by the program, followed by a menu of the possible device types.
DEV=FSS	Device-type is 8" Floppy Single side, Single density.
DR=FPY1:	The drive where the disc is.
V=FMST	Gives the disc the name ´FMST´.
CLEAR	Clear and readcheck of the disc.
START	Start the initialization.
No sectors flagged	When finished, the program presents a table of the disc characteristics.
END	Terminate the program.
-12.34.56 End-of-task 0	
-CLOSE FPY1:	Close the drive.
-OPEN FPY1:	Open the drive filestructured, if You want to use the disc immediately.

DISK BOOTSTRAP GENERATOR (BOOTGEN)

5.1 INTRODUCTION

The Bootstrap Generator, BOOTGEN, is used to write a loader sequence of 1 KB on the sectors 1 to 4 on a disc device. This loader sequence, which is booted into the memory by the System Bootstrap Loader, then loads the final program to be started, normally the Operating System.

5.2 SYSTEM REQUIREMENTS

The BOOTGEN program requires:

-6 KB of main memory above the operating system.

-a console device.

-a currently supported disc device containing the program to be started.

5.3 STARTING

The following procedure is recommended to write the loader on the disc:

-Open the disc-device NON-FILESTRUCTURED by the command:

```
OPEN,N dev:
```

-Start the program with the command:

```
BOOTGEN,t dev:filename
```

<t>, type of disc:
M, mini floppy interface, 4034. This is the default type.
F, floppy interface, 4076
B, one-board floppy controller, either 4106 or 4108.
H, hard disc interface, 4109.

<dev>, name of the disc device, NOT volume !

<filename>, name of the program to be loaded and started.

5.4 MESSAGES

Possible messages are:

BOOTGEN Rx.yz
Signon at program start, where the revision level is <x>, and the update level is <yz>.

PLEASE RELOAD PROGRAM, NOT RESTARTABLE
Impossible to restart the program without reloading it first.

NO PARAM.
Start parameter missing.

INV. NAME
Syntax error, or device/filename missing.

DISK-ASGN
Failed to assign the disc device.

DISK-READ
Read error either on sector 0, or in the directory.

NOT FOUND
Filename not found in the directory.

DISK-WRITE
Write error on one of the sectors 0 to 4.

END OF TASK n
Where <n> may be the SVC error status.

5.5 EXAMPLE

This is the normal sequence to write down a loader:

-CLOSE FPY1:	Close the drive if it is opened file-structured. Then mount the disc to be bootgened.
-OPEN,N FPY1:	Open the drive non-filestructured.
-BOOTGEN,F FPY1:OSNAME	Load and start the program, and pass start parameters to it.
Bootgen Rx.yz	Signon from the program.
Doing floppy boot !	Tells the type of boot written down.
-12.34.56 End-of-task 0	Good task termination.
-CLOSE FPY1:	Finish-up by closing the drive. It is now possible to boot from the disc.

DISK INTEGRITY CHECK (DISKCHECK)

6.1 INTRODUCTION

The Disc Integrity Check, DISKCHECK, provides a means of recovering open disc files following an operating system crash. The program closes all files found to be assigned, and validates some control information on the disc.

6.2 SYSTEM REQUIREMENTS

The DISKCHECK program requires:

-4 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

6.3 STARTING

The following procedures are recommended after an operating system crash for systems configurated with direct-access devices:

-Reload the operating system.

-Mount the disc to be checked.

-Open the disc-device NON-FILESTRUCTURED by the command

OPEN,N dev:

-Start the program with the command

DISKCHECK dev:(,list)

where <dev> is the name of the device holding the disc, and <list> is an optional list device.

-Close the disc-device.

-It is now possible to open the disc file-structured.

6.4 MESSAGES

The program may output the following messages:

DISKCHECK Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

PLEASE RELOAD PROGRAM, NOT RESTARTABLE
The program must be loaded prior to each new session.

NO PARAM.
Start parameter missing.

INV.NAME
Syntax error in file-descriptor or device name missing.

DISK-ASGN
Failed to assign the device(s).

DISK-RE/WR
Directory read/write error.

ERROR IN HASH KEY SECTOR IN tFD DIRECTORY
Invalid data in a type <t> directory.

RE/WR ERROR IN tFD DIRECTORY
Read/write error in a type <t> directory.

<filename> HAS NO FIRST INDEX SECTOR, IS DELETED
A file only present with it´s name in the directory is deleted.

<filename> WAS ASSIGNED FOR READ/WRITE
An open file is closed.

END OF TASK s
Where <s> is the SVC-error code.

6.5 EXAMPLE

This example shows the standard procedure to close opened files:

```
-CLOSE      FPY1:                 Close the drive if it is opened. Then
                                  mount the disc to be checked.
-OPEN,N     FPY1:                 Open the drive non-filestructured.
-DISKCHECK FPY1:                  Start the program and pass drive as
                                  start-parameter.
Diskcheck Rx.yz                   Signon from the program.
<filename> was assigned for read  Loggs a message for each opened file.
-12.34.56 End-of-task 0           The program terminates good.
-CLOSE      FPY1:                 Close the drive, and
-OPEN       FPY1:                 open it for normal use.
```

DISK DUMP (DISKDUMP)

7.1 INTRODUCTION

The Disc Dump utility, DISKDUMP, provides a facility for displaying the information on a disc volume, or in a disc file, in a format useful for debugging system and user routines.

7.2 SYSTEM REQUIREMENTS

The program requires:

-4 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

7.3 STARTING

The program is started by the command:

DISKDUMP (cmdinput)

If the <cmdinput> file-descriptor is missing, the program enters the interactive mode, and prompts on the console for command. If the <cmdinput> file-descriptor is specified, the program enters the remote mode. If any error occur in the remote mode, the program terminates with the error status as End-Of-Task code.

7.4 COMMANDS

The function of the program is controlled by commands, that could be entered either in interactive or remote mode.

If <count> is omitted in any dump command, a default count equal to one is used. If <sect> and <count> are omitted, then the next sector in sequence is dumped.

INPUT FD
Assign data input to <fd>.

DUH SECT(,COUNT)
Dumps <count> sectors starting from <sect> in hexadecimal format.

DUO SECT(,COUNT)
Dumps <count> sectors starting from <sect> in octal format.

DUA SECT(,COUNT)
Dumps <count> sectors starting from <sect> in ASCII image format, pure data only. Can be used to dump text files from non OS.8 discs.

LIST ON FD
Turns on listing copy option to <fd>.

LIST OFF
Turns off listing copy option.

SKEW
Turns on skew read flag if skewed read is supported by the driver.

UNSKEW
Turns off skewed read.

END
Exit from program.

7.5 MESSAGES

The program may output the following messages:

DISKDUMP Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

BAD COMMAND DEVICE
Failed to assign the command input.

UNKNOWN COMMAND
When an unrecognized command is entered.

PARAMETER ERROR
Missing or bad parameter.

ASSIGN ERROR <s>Q, NAME <filename>
Error status <s> when failed to assign the input <filename>.

DISKERROR: FC=<f>Q, STAT=<s>Q
Input error <s> at function code <f> on the input device.

END OF TASK <s>
Where <s> is the SVC error status.

7.6 EXAMPLE

This example shows how to examine a file, perhaps generated by a user program:

```
-DISKDUMP                  Load and start the program.
Diskdump Rx.yz             Signon from the program.
IN MYFILE                  Assign the file to be examined.
LIST ON PR:                Assign list device for a copy.
DUH 0,2                    Dump in hexadecimal, on the console and on
                           the list device, the first two sectors in the
                           file.
END                        Exit the program.
-12.34.56 End-of-task 0    Program terminates.
```

COPY UTILITIES

8.1 INTRODUCTION

This chapter describes four different useful copy utilities, each explained in its own part:

COPYLIB, copy/delete under directory control.

COPYA, copy ASCII data.

COPYT, copy a task.

COPYI, image copy.

COPYLIB UTILITY (COPYLIB)

8.2.1 INTRODUCTION

The COPYLIB utility provides a fast method of saving files. The files may be copied from volumes to either volumes, or files, or devices. This is done by reading the content of the directory on the source volume.

The transfer takes place without any regard to the type of the file, or the content in the file, except transfer of ASCII files to a printer.

The COPYLIB program may also be used to delete files under directory control.

The program may, depending on the start switches, execute either in an interactive mode, or in a remote mode. It is possible to use wild-card specification of filenames, as well as a selection file.

The interactive mode works in two phases. The first is a questioning phase of what to do with each file, and the second is the real working.

8.2.2 SYSTEM REQUIREMENTS

The program requires:

-10 KB of memory above the operating system, including 1 KB buffer.

-a console device.

-a currently supported disc device.

8.2.3 STARTING

The program requires some start parameters:

COPYLIB(,<D><G>(,BUFSIZE)) SOURCE(,(DESTINATION)(,SELECTFILE))

The switch <d> is used when the program should work with the delete function.

The switch <g>, as in ´GO´, is used to enter the remote mode.

The additional <bufsize> expands the copy buffer, which increases the speed of the copy phase.

The <source> specifies the input and/or delete volume, where wild-card file-name specification may be used.

The <destination> is the name of the volume, or the device, or perhaps the Element-File Directory, to which the files shall be copied.

The <selectfile> is the name of a file or a device, that contains the names of the files to be copied or deleted. The <selectfile> is an ASCII-file, containing one filename or a wild-card specification on each line. When a <selectfile> is specified, the program enters the remote mode.

8.2.4 COMMANDS

When the program is entered in the interactive mode, a menu of the present commands is displayed, and they are:

A	Abort execution, don´t copy/delete.
C(=FILE(.ELEMENT))	Copy, perhaps using new name.
D	Delete from source.
I	Ignore the rest of the directory, start working.
P	Pause the execution.
´CR´	Hit the ´RETURN´ key to skip the file.

A question is then raised for each found file, and one of the commands above should be given as an answer.

8.2.5 MESSAGES

The program may output the following messages:

COPYLIB Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

NOT RESTARTABLE, RELOAD PROGRAM
It is impossible to restart the program without reloading it first.

ASSIGN ERROR
Failed to assign input or output.

SELECT FILE ERROR
Failed to assign or read from the select file.

INPUT OR OUTPUT NOT DIRECTORY ORIENTED
Invoked device of wrong type.

TABLE FULL
The interactive response table full.

ERROR AT READ
When failed to read.

ERROR AT WRITE
When failed to write.

END OF TASK <s>
Where <s> is the SVC error status.

8.2.6 EXAMPLE

The examples shows some different situations of the copy and the delete functions:

COPYLIB FMST:,FCPY:
Each file on the volume FMST: will be presented to the operator, and a question of what to do is raised. If <c> is specified, that file will be copied to FCPY:. If <d> is specified, that file will be delete from FMST:.

COPYLIB,DG FMST:*AB-
Each file containing ´AB´ in the second and the third position will be deleted without answering any question.

COPYLIB,G,14000 FMST:OWNDIR.AB-,FCPY:
Each element beginning with ´AB´ in the Element-File OWNDIR on the volume FMST:, is copied to the volume FCPY:. File-names will be the same as the original element-names.

COPYLIB,G,14000 FMST:FILE,FCPY:OWN
If FILE is an Element-File Directory, the elements are copied into the Element-File Directory OWN on the volume FCPY:. If FILE is a normal file, the first file in the FMST:-directory named ´FILE´, is copied to the volume FCPY:, and given the name OWN.

COPYLIB,DG FMST:-/A
Delete all ASCII-files found on the volume ´FMST´.

COPYLIB FMST:,FCPY:,SELECT
Copy all files specified in the file ´SELECT´, from the volume ´FMST´ to the volume ´FCPY´.

COPY ASCII UTILITY (COPYA)

8.3.1 INTRODUCTION

The COPYA utility is used to copy ASCII data between devices and/or files. There are some switches to control the data formatting and access methods.

8.3.2 SYSTEM REQUIREMENTS

The program requires:

-2 KB of memory above the operating system, including 0.25 KB buffer.

-currently supported devices.

8.3.3 STARTING

The program requires some start parameters:

COPYA(,<A><I><R>) SOURCE,DESTINATION

The switch <a> is used to append the sourcefile to the destination.

The switch <i> indicates that the data sould be transferred as ASCII image data. The default is formatted transfer, refer to OS.8 Program Reference Manual.

The switch <r> makes the transfer to take place as random access, to be used to copy formatted ASCII files with positioning information. Default is sequential access.

The <source> specifies the input file or device.

The <destination> is the name of the output file or device, created if not existing.

8.3.4 MESSAGES

The program may output the following messages:

ASCII COPY Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

NEW FILE
If the destination was created.

ERROR <s> ON INPUT LU 0
    Failed to assign to, or read from, the source. The SVC error status is <s>.

ERROR <s> ON OUTPUT LU 1
    Failed to create, or assign/write to the destination. The SVC error status is <s>.

END OF TASK 0
    The program terminates.

COPY TASK UTILITY (COPYT)

8.4.1 INTRODUCTION

The COPYT utility is used to copy task files between devices and/or files. The program can copy both relocatable and absolute task files.

8.4.2 SYSTEM REQUIREMENTS

The program requires:

-6 KB of memory above the operating system, including 4 KB buffer.

-currently supported devices.

8.4.3 STARTING

The program requires some start parameters:

COPYT SOURCE,DESTINATION

If the <destination> is a file and already present, it will be deleted and created again. This to allocate the exact amount of space, to be able to handle absolute contiguous files.

8.4.4 MESSAGES

The program may output the following messages:

COPY TASK Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

BAD INPUT FILE DESCRIPTOR
Syntax error in source name.

BAD OUTPUT FILE DESCRIPTOR
Syntax error in destination name.

INPUT ASSIGN ERROR <s>
Failed to assign the input file, where <s> is the error status.

INPUT FORMAT ERROR
The input is not a task file.

OUTPUT ASSIGN ERROR <s>

Failed to create and assign the output file, where <s> is the error status.

OUTPUT ERROR
Failed to write to the output.

INPUT ERROR
Expecting more data, didn´t find end-of-file or timeout.

RECORD SIZE ERROR
Inputted data not modulo 256 bytes.

<nnnnn> RECORDS COPIED
Number of records copied.

END OF TASK 0
The program terminates.

IMAGE COPY UTILITY (COPYI)

8.5.1 INTRODUCTION

The COPYI utility is used to copy and/or verify data between devices and/or files. There are some switches to control the action of the program.

8.5.2 SYSTEM REQUIREMENTS

The program requires:

-5 KB of memory above the operating system, including 2 KB buffer.

-currently supported devices.

8.5.3 STARTING

The program requires some start parameters:

COPYI(,<V><O>(,BUFSIZE)) SOURCE,DESTINATION

The switch <V> is used to verify the sourcefile with the destination, which is done after the copy job.

The switch <O> is used together with the switch <V>, and indicates that no copy shall be done, verification only.

The additional <bufsize> expands the copy buffer, which increases the speed of the copy and/or the verify phase.

The <source> specifies the input file or device.

The <destination> is the name of the output file, created if not allocated, or the output device.

8.5.4 MESSAGES

The program may output the following messages:

IMAGE COPY Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

NEW FILE
If the destination has to be created.

ERROR <s> ON INPUT LU 0
    Failed to assign to, or read from, the source. The SVC error status is <s>.

ERROR <s> ON OUTPUT LU 1
    Failed to create, or assign/write to the destination. The SVC error status is <s>.

COPY REQUESTED
    Indicates start of the copy phase.

<nnnnn> RECORDS COPIED
    Number of 256 bytes records copied.

VERIFY REQUESTED
    Indicates start of the verify phase.

BYTE <S> NOT EQUAL TO <C> IN RECORD <R> AT <A>
    Verify mismatch between source byte <s> and verified byte <c> in record number <r> at relative address <a>. Each record is 256 bytes, and the address is within a record.

VERIFY OK
    When no mismatch was found between source and destination.

END OF TASK 0
    The program terminates.

OBJECT LIBRARY EDITOR (OBJLIB)

10.1 INTRODUCTION

The Object Library Editor, OBJLIB, is used to manipulate with Object Library Files. The program will interactively accept commands and respond with messages to the console operator. Through available commands, an Object Library File can be created on a bulk storage device. This file may be searched for a particular module, selectively copied to another I/O-unit, or added to another under operator control.

10.2 SYSTEM REQUIREMENTS

The program requires:

-9 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

10.3 STARTING

When the program is started by the command

OBJLIB

it enters the interactive mode, and prompts on the console for commands. The logical unit 0 is assigned to the console by the program.

10.4 COMMANDS

The function of the program is controlled by commands, that is entered in interactive mode:

ASsign LU,FD
: Assign logical unit <lu> to <fd>. The default file-type is OBJ. The file will be created if it is nonexistent. Logical unit 0 is assigned to the console.

CLose LU
: Close the file presently assigned to logical unit <lu>.

COpy LUI,LUO(,MODULE)
: Causes one module to be copied from <lui> to <luo>. If a <module> is specified in the command, the <lui> is first

searched for the <module> as described under FIND. If no <module> is specified, the first module encounted on the <lui> is copied. After a COPY operation, the logical units are positioned at the end of the module copied.

DUplicate LUI,LUO(,MODULE)
Causes module(s) to be copied continuously from <lui> to <luo> until <module> is found. The specified modlue is not copied and the operation is terminated. If no <module> is specified, or not found, module(s) are copied until end-of-file on <lui> is detected.

ENd
Terminate program.

FInd LU,MODULE
Causes <lu> to be searched until <module> is found. The unit is then backspaced to the beginning of the found module. The module may now be copied. The <lu> is not rewound when the FIND command is given. This command should be used to position a file containing several modules.

PAuse
Will set the program in pause state.

RWind LU
Rewind the file assigned to <lu> to beginning of file.

TAble LU,LUL
Scans the <lu> and lists on <lul> all module names found from the position of the file when this command was given. To create a table of all module names the user must issue a rewind command, or know that the file is at the beginning. Thus a table of the content of a file containing many modules may be obtained.

10.5 MESSAGES

The program outputs the following messages:

OBJLIB Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

???
When an unrecognized command is entered.

EOF
When end-of-file is detected.

NEXT MODULE IS:<MODULE>
Tells the name of next module after a copy, duplicate or find command.

READ ERROR LU <n>
Abnormal read-status on logical unit <n>.

I/O ERROR
Write failed.

FILE OPEN ERROR, CODE:<s>
Failed to assign and/or allocate a file, where <s> is the error status.

END OF TASK <s>
The program terminates, and where <s> is the SVC error status.

10.6 EXAMPLE

The following example shows the use of the Object Library Editor for library manipulation:

-OBJLIB	Load and start the Object Library Editor.
OBJLIB Rx-yz	Signon from the program.
>AS 1,OLDLIB	Assign and position to beginning of the old master.
>AS 2,CURLIB	Assign current master.
>AS 3,NEWLIB	Assign, perhaps allocate, new master.
>DU 1,3,SUBR	Duplicate all modules up to module SUBR, from old master to new master.
>FI 2,SUBR	Find and position to module SUBR in current master.
>CO 2,3	Copy one module, SUBR, from current master to new master.
>RW 3	Rewind new master.
>TA 3,0	List the content of the new master on the console.
>EN	Exit program.
-12.34.56 End-of-task 0	Program termination.

TASK ESTABLISHER (ESTAB)

11.1 INTRODUCTION

This chapter describes the OS.8 Task Establisher, ESTAB. Any task must be established using ESTAB before it can run under OS.8.

The functions of ESTAB and the commands used to control it, are fully described in this document. The user should reference the OS.8 Program Reference Manual for detailed information about task preparation within an OS.8 environment.

A task may be a single program or a group of programs linked together. ESTAB processes object-code programs, links external references, and produces either a relocatable task for loading and running under OS.8, or an absolute memory-image stand-alone task such as the OS.8 itself.

The command stream directing ESTAB activity can be input in remote mode or interactively. An operator uses the commands to specify programs for inclusion in the task, as well as task options. The establishment procedure requires two passes of the object code. In the first pass, ESTAB collects the modules to be included, and compiles a symbol table of external references and definitions. In the second pass, the actual Task is built.

11.2 SYSTEM REQUIREMENTS

ESTAB requires 15KB of memory space, and as much space as is required to house a dictionary of all external references and definitions in the programs of the task being established. ESTAB builds task modules using disc file as work storage. Required devices include input and output binary discs for the input object code and output relocatable or absolute code. A temporary disc file is allocated on the system volume to hold pass one input programs for use during pass two. ASCII devices requirements are one for the command input and another for the list output.

11.3 STARTING

The generalized start command format is:

```
ESTAB(,,ADDMEM) (INPUT)(,(OUTPUT)(,(RADIX)(,LIST)))
```

The <ADDMEM> is the additional memory size in bytes, that should be added to the ESTAB to hold the symbol table. A default table sufficient for 100 symbols is contained within the ESTAB.

The <INPUT> file specification determines the program operating mode. If it does not exist, the program will enter the command mode, and prompt at the console for commands. If an input file is specified, the program will enter either the direct mode or the command mode. The direct mode is entered if the file is in OBJECT format, and that file will be used as the only input file during the task establishment, and no command processing will take place. If the file is in ASCII format, the command mode will be entered. The command mode can also be entered if the prefix "CMD=" is typed in front of the input file specification.

The <OUTPUT> file specification will cause creation and assignment of an output file.

The <RADIX> is "O" or "H" and sets the values in the listing to be written in Octal or Hexadecimal form. Default is Hexadecimal.

The <LIST> determines where and if the linking list should be written.

All parameters above may be overidden by a command in the command mode.

11.4 EXPRESSION HANDLING

All values can be entered as expressions, which are evaluated from left to right on a strict 16-bit integer basis. Operands can be either symbols already defined, or numerics such as octal, decimal, hex values and ASCII-strings. Legal operations are +(PLUS), -(MINUS), /(DIVISION), *(MULTIPLICATION), &(AND), !(OR), ?(XOR) and #(SHIFT). Note that only the PLUS and MINUS operations are allowed on non-absolute values. Note also that some commands require absolute values, such as ORG and PLCBASE. The value of the PLC selected by the PLCNR command can be addressed by the symbol '*', where this value never is absolute.

11.5 GENERAL

The following shortenings are used through the section:

FOMD
: Standard File Or Module Description.

SYMBOL
: Any legal symbol, see Assembler Manual.

VALUE
: An expression that can be evaluated to a value.

PLCNR
: An expression that can be evaluated to a value representing a PLC. The value must be between 0 and 31, where 0 represents PLC 0.

OPT
File search options used together with the INC and LIB commands:
A, define the unsolved absolute symbols, do not include the module.
E, search through the file until End-Of-File is found.
R, (default) the file is searched from the current position. When End-Of-File is found, the file is rewound and searching resumed. This process continues until either no unsolved references exist, or no more references can be solved in that file.
W, same as "R", but the file is rewound before searching is started.

11.6 COMMANDS

This section describes the commands that can be given. A command line starting with an asterisk '*' in the first column is handled as a comment.

ABOrt
Program execution is aborted. The output file is deleted.

ABSolute
Generates an absolute task. Default is relocatable.

CHAin FOMD
Re-assign the command file.

CHEck
Lists unsolved or multiple defined references on the console, with the location first referenced.

CEQu VALUE,SYMBOL(,SYMBOL...)
Same as EQU with the exception: If the symbols already have a value, that value is not changed, and no error occurs.

CEXternal VALUE
Gives the VALUE to those CEXTRN symbols that are referenced but not given any value through the linking.

ENd (VALUE)
End of command sequnce. The output file and the listing are produced. If VALUE is present, it defines the program start address.

EQu VALUE,SYMBOL(,SYMBOL...)
The SYMBOLs listed are given the specified VALUE. If the symbols are not found, they are entered into the symbol table. If the symbols already have a value, they become multiple defined.

EXpand VALUE
Is used to add additional memory to a task beyond what is required to hold the code body. VALUE is the number of bytes to be added.

INclude(,OPT) FOMD(,FOMD...)
Will include a file or a module into the the linking. If a filename is given, the old file, if any, is closed and the new file is assigned. If a module name is given, the file assigned will be searched for the module and the module will be included. If no module is given, the whole file is included. The search and inclusion depends on the options given.

LIBrary(,OPT) FOMD(,FOMD...)
Will search through the file or module and look for any ENTRY that corresponds to an undefined symbol. If so, the module is included. If a module is given, the file will first be searched for the module, and then the module will be searched for entries corresponding to undefined symbols. If no module is given, each module in the file will be interpreted. The search through the file, depends on the option given. If E option is given, the file must be ordered in such a way, that a module only will refer another module appearing later on in the file.

LISt/NOList (ABSolute/UNUSed)
Will enable/disable the listing of either all symbols, or either all symbols with an absolute value, or all symbols not referenced during the linking.

LOg FD
Directs a log of the commands and the error messages to a log file. Each INC and LIB command will be printed separatly on the linking list.

MAXLu n
Is used to set the upper limit of the number of disk files the established task can use. Default <n> is 3.

MAXNode n
Sets the number of nodes that will be generated to the task at load time. Default <n> is 8.

OBject FD
Renames the temporary file and saves it. This is useful to create a new object library from others, using the LIB and INC commands.

OPtion opt(,opt...)
Set task load options:
RESident, makes task memory resident.
NONAbortable, not abortable from other tasks.
DEFAssign, default Logical Unit assignment.
NOStackcheck, disables stack limit check at SVC.
ERMsg, error messages generated by operating system.

ORG VALUE
Will origin the PLC given in a previous PLCNR command.

PAuse
Will put the Task Establisher in pause state.

PLCBase VALUE
Orginates the first PLC in a sequence defined by a PLCOrder command.

PLCCommon plcnr
Selects the <plcnr> under which any common-areas will be allocated. Default PLC is 1.

PLCEnd
Generates the symbols $$PLCE00 through $$PLCE31. These symbols, that may be referenced in the user program, will, after the END command, be given the value of the first free location after respective Program Location Counter.

PLCList plcnr(,plcnr)
Selects the PLC(s) to be printed in the linking list. Default is PLC 0 and 1.

PLCNr plcnr
Selects a PLC for a subsequent ORG command.

PLCOrder plcnr(,plcnr...)
Enables the user to specify in which order the PLC-segments are to be putted into memory.

PLCStart
Generates the symbols $$PLCS00 through $$PLCS31. Each symbol is given the start value of each PLC, which is controlled by the PLCBase command.

PRINt FD
Specifies where the linking list should be written. This command will overide the <FD> given in the start command.

PRIOrity prio
Sets the default priority to a task. The priority must be an absolute value and in the range 9<prio<250. Default priority is 128.

RADix 8/16
Selects the base to be used when printing values.

REDefine VALUE,SYMBOL(,SYMBOL...)
Will change each already defined SYMBOL to the new VALUE.

RELocatable
Will generate OS.8 Relocatable Task File as output. This is default.

REMote
Causes the linking to be aborted, if an error is detected.

SELect SYMBOL(,SYMBOL...)
Enters the SYMBOls as undefined into the symbol table.

STACKlimit VALUE
Will set the task load stack size. Default is 256.

TAsk FD
Specifies the output task file.

UNSolved VALUE
Gives VALUE to those symbols that are referenced but not given any value through the linking.

VERsion X.YZ
Sets the version number in task files. Default is 0.00.

11.7 MESSAGES

The following messages may be written by the ESTAB:

ESTAB R.xyz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

RELOAD PROGRAM
Is written if the program is restared without being reloaded.

CONSOLE ASSIGN ERROR
Failed to assign the console.

COMMAND ERROR
Undefined command.

COMMAND NOT YET IMPLEMENTED
A command to be implemented in the future.

PARAMETER ERROR
Parameter missing or invalid expressed.

THAT SYMBOL IS ALREADY DEFINED
Tries to define an existing symbol.

FILE NAME ERROR
Syntax error in a file-descriptor.

FILE NOT FOUND
File or device not present.

INPUT FILE NOT SPECIFIED
Tries to include a module without having an input file assigned.

MODULE <name> NOT FOUND
Module not present in the file.

END OF TASK <s>
Program terminates, and where <s> can be: 0-No errors, 1-Undefined or Multiple Defined Symbols, 2-Aborted.

11.8 EXAMPLE

This example illustrates how to establish a simple task. The object library file MYLIB contains the object module(s):

```
-ESTAB                      Load and start the task establisher.
Estab Rx.yz                 Signon from the program.
TASK    MYTASK              Assign, perhaps allocate too, the result.
INCLUDE MYLIB               Include the module(s).
PRINT   PR:                 Produces a map of the task on the printer.
END                         Generate the result, and exit.
-12.34.56 End-of-task 0     Terminates with good status.
```

TASK TRACE (TRACE)

12.1 INTRODUCTION

The Task Trace Utility, TRACE, is an interactive utility program designed to assist the user in debugging tasks in the assembly language level in an Operating System environment. It has powerful facilities for monitoring the execution of a program, either in real-time or interpretive mode, and allows modifications to be made easily at run time.

The TRACE program is available in a disk version supported by an operating system, or in a stand-alone version.

12.2 SYSTEM REQUIREMENTS

The program requires:

-9 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported load device.

12.3 STARTING

When the program is started by the command

TRACE(,D/H/O) (fd(,sw) (par1(,par2...)))

it enters the interactive mode, and prompts on the console for commands.

The optional switches D, H and O is used to set up either decimal, hexadecimal or octal as the default radix. If omitted, hexadecimal radix is chosen.

The optional parameter <fd> is the name of a file containing the program to be traced. This program is loaded and the debug session is started automatically. The optional switches <sw> and parameters <par1> and <par2...> are passed to the started task.

12.4 ENVIRONMENT

There are no demands or requirements on a program that should be traced, and it is possible to debug interrupt routines with the stand-alone version of the TRACE.

If the TRACE is started without parameter, the first thing to do is to load the program that should be debugged. The second is to start the program, with or without switches and parameters. The TRACE task will then reside in memory in a paused state and be used as a run-time library by the debugged task.

When the debug session is finished, the TRACE task may be removed with the CANCEL command, or continued with the CONTINUE command. In that case the TRACE will prompt the user for the next command.

12.5 OPERATING MODES

The TRACE works in two modes, a Command Mode and a Trace Mode. The Command Mode is used to prepare for the debugging session, and the Trace Mode is the real debugging.

The Command Mode handles the debug preparation such as loading and starting the program to be traced, specifying trace conditions and manipulating memory and register contents etc.

The program execution is controlled in the Trace Mode by single-key commands, which also allows the user to specify certain trace condition.

12.6 COMMAND MODE

A large set of commands are available in the command mode. commands controls the futher action of the TRACE, and allows the to specify breakpoint addresses and thier actions. The user is so supported with the possibility of a copy function and user defined symbols.

12.6.1 EXPRESSION HANDLING

All values can be entered as expressions, which are evaluated from left to right on a strict 16-bit integer basis. Operands can be either symbols already defined or numerics such as octal, decimal, hexadecimal values and ASCII characters. Legal operations are +(PLUS) and -(MINUS). The current program location can be refered to with the symbol '*', and the last inputted address value can be refered to with the symbol '#'. Numeric values are expressed in the default radix. Some commands have switches that allows the use of another radix than the default radix.

12.6.2 TASK COMMANDS

These commands are related to task functions:

ENd
: Terminates the debugging. Don't forget to remove the TRACE task.

LOad fd(,(tid)(,addmem)
: Load a task from <fd>. The optional parameter <tid> will be the name task, and if omitted, it is set to the first four

characters in <fd>. The <addmem> is the amount of area that sould be added to the task at load time.

PAuse
The user program is paused. The command COntinue TID should be used to resume the debugging, where <tid> is the name of the user program - not the TRACE identity USPn !

STart,(switches) tid(,parameter)
Start debugging the task named <tid>, and pass <switches> and <parameters> to it. The task is started at its ordinary entry address on which a single-step breakpoint is inserted. Then the TRACE task pauses itself and the debugging continues under the started program identity, which uses the TRACE program code as a subroutine package.

12.6.3 GENERAL COMMANDS

This is a command group that contains general pupose commands:

CONvert value
Convert and dispaly the <value> to both decimal, octal and hexadecimal values.

COPy ON fd
Enables the copy function to <fd>. All commands and informations displayed on the terminal is copied to <fd>.

COPy OFf
Disables the copy function.

RADix 8/10/16
Selects the default base for numeric values.

SET name=addr
Set a symbolic relocation register <name> to the value <addr>. The <name> is a user defined symbol, not exceeding 6 characters length, which may be used in any expression. In hexadecimal radix the name must not conflict with a hexadecimal number such as AE or BED etc.

SVc OFf/ON
Disables/enables the interpretaion of the RST 7 instruction as a SVC. Default is enabled.

12.6.4 MEMORY COMMANDS

This group of commands are used to manipulate the content of main memory:

EXamine(,A/D/H/O) addr(,size)
Displays the memory content in either ASCII (A), or decimal (D), or hexadecimal (H), or octal (O). The display starts at the address specified by <addr>, and <size> bytes are displayed. If <size> is omitted, 1 byte is displayed.

FILL(,D/H/O) addr,size,value
Fills a memory area <size> bytes large starting at <addr> with <value>. The radix of the <value> is defined by the optional switches and may be either decimal (D), or hexadecimal (H) or octal (O).

MOdify addr
Displays 16 bytes of memory starting at address <addr>, the operator may now modify the content of the memory cells by typing the new values separated by a comma and terminated by a return. If only return is entered nothing is modified.

12.6.5 CPU COMMANDS

In order to support the user with the possibility to control the Central Processor Unit, this set of commands has been implemented:

DISable
Disables the interrupt logic of the CPU. Use only with the stand-alone version.

ENAble
Enables the interrupt logic of the CPU. Use only with the stand-alone version.

INput port
Displays the content in the I/O-port with the address <port>.

NMI
Enables the TRACE to take care of non-maskable interrupts, which will act as a breakpoint when it occurs.

OUT port,value
Sends the data <value> to the I/O-port with the address <port>.

REGister (register=value)
Display the content of the current register set if the parameter is omitted. Change the content of the <register> to the <value>. The registes are specified by a mnemonic:

A = byte register A.
F = condition code, where the flags are abbreviated:
Z = zero true
C = carry true
S = sign true
P = parity true
To clear all flags enter REG=return-key.
BC = register pair BC.
DE = register pair DE.
HL = register pair HL.
X = index register X, 16-bit.
Y = index register Y, 16-bit.
SP = stack pointer, 16-bit.

12.6.6 BREAKPOINT COMMANDS

Breakpoints are user-selected locations at which the program execution is halted to permit interaction between the user program and the TRACE. Breakpoint facilitate free-run execution of the program. A breakpoint is furnished by a RST-instruction (normally RST 1), which is placed at any instruction location in the user program. The number of breakpoints is only limited by the amount of memory added to the TRACE at task establish time. When a breakpoint is reached, the TRACE checks that the location belongs to an adequate instruction.

During an active debugging session no breakpoints are placed in the user program at single-step or interpreting mode. When the trace mode changes to free-run, all breakpoints will be stored into their locations and the control is transferred to the user program. The breakpoints will be removed when a breakpoint is encountered, and the original instructions will be restored.

This chapter describes the commands which is used to control the use of breakpoints.

Halt addr
: Stops the program execution at <addr>.

JTAble addr
: Specifies a base address for a jump-table containg 256 bytes. The address is always at a 256 byte boundary. All programmed entries through this table will not be traced.

LIst
: Displays the current trace conditions and their addresses.

REmove (addr)
: Remove all trace conditions if the parameter is omitted, or the breakpoint at the address <addr>.

SUBroutine addr
: Defines a subroutine, by its entry address <addr>, not to be traced. Subroutines may be debugged in an earlier stage and excluded from being traced when debugging program modules. When the defined subroutine is called from the user program, a message is printed and the subroutine is executed in free-run. Refer to the warning described with the trace command ´X´ !

TRace(,D/J/M/R/S) addr
: Put a trace condition at the address <addr>. The trace condition is specified by the switch:

  Omitted
  : Enter the single-step mode.

  D(isable)
  : Enter the free-run mode. The program execution will continue until a new breakpoint is encountered.

  J(ump)
  : Enter the interpreting mode. Print only executed JMP, CALL and RET instructions.

M(emory)
Enter the interpreting mode. Print the change of any memory cell defined by the WATCH command at breakpoint time.

S(can)
Enter the free-run mode. Print every change of any memory cell defined by the WATCH command.

R(egister)
Enter the free-run mode. Print every change of any CPU-register at breakpoint time.

WATch addr(,addr...)
Define an address in memory to be watched for any changes. The defined memory cell will be printed if changed. The printing occurs both in the interpreting mode, and in the free-run mode when a breakpoint is reached.

12.6.7 DEBUG COMMANDS

These commands are used to enter the Trace Mode, where the program execution is controlled:

DEBug addr
Start debugging at the address specified by the <addr>.

PROceed
Continue the debugging at the last program location when the command mode was entered.

12.7 TRACE MODES

This mode works in two sub-modes of the real debugging session, the Interpreting Mode and the Free-run Mode.

In the Interpreting Mode the execution of the user program is performed by TRACE and not by the user himself. The execution takes place through an emulating technique, where the TRACE acts in the same way as the Central Processor does.

In the Free-run Mode the control is transferred to the user program, and if neither any breakpoints are specified, nor a breakpoint is reached, the user program never returns to the TRACE.

The Trace Mode is identified by the cursor position, which is located at the same line as the instruction that is to be executed. The cursor is placed immediately after the instruction, and the action is controlled by trace commands.

When the control is returned to the TRACE, it always displays any register changes and the next instruction to be executed. The instruction is displayed by its mnemonic, and indirect addresses are displayed with their absolute values.

12.7.1 TRACE COMMANDS

This chapter describes a set of one-key commands that are to be used in the trace mode: SPACE KEY
Step to next instruction in single-step mode.

D(isable)
Enter free-run mode at the current location.

J(ump)
Set a breakpoint at the target address location.

K(ill)
Remove the breakpoint at the current location, if any.

L(oop)
Execute the current DJNZ instruction until end. May be used at any two bytes instruction. Eg put a temporary breakpoint at current program location+2.

R(egister)
Print all CPU-register content. Does not affect any register or the trace mode.

T(race)
Specify a single-step breakpoint at the current location.

W(atch)
Specify the current address argument to be checked for changes.

X(ecute)
Supress the currently entered subroutine. The return address must be last item push on the CPU-stack.
WARNING !
Note that the top of stack is not the address to return in the user program, but an address in the TRACE program to catch the next return instruction. If the top of stack item is used as a parameter, dont use ´X´ or SUBROUTINE or JTABLE commands !

ANY OTHER KEY
Exit trace mode and enter command mode.

12.8 MESSAGES

The program outputs the following messages:

TRACE.8 Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

NO I/O-RAM
Informs that no I/O-RAM is present on the controller card.

ERR-TYPE:tttt, POS:pppp
When either an invalid command, or parameter or switch is entered. The position field <pppp> describes where in the command line the error occured, and the type field <tttt> contains a detailed explanation:

COMMAND, when an unknown command is entered.
PARAM., when an invalid parameter is entered.
OPTION, when a non-existing switch is entered, or cant be satisfied.
NAME, the file descriptor is not found or invalid.
TASKID, the specified task is not found or invalid.

END OF MEMORY
End of memory when defining a symbol in the SET command.

TAB-OVF
Table overflow. Can´t specifiy any more trace locations.

CAN´T FIND IT
The address specified in a REMOVE command is not found.

ILLEGAL BREAKPOINT
A RST-instruction occured at a non specified address. Start the debug session from the beginning.

addr WAS oo IS nn
The memory cell at address <addr> has been changed. The original content was <oo> and the new content is <nn>.

SUBROUTINE RUN..
An excluded subroutine is entered.

TRACE STOP
A previous defined HALT location is reached.

HALT TRACED
The CPU-instruction HLT is discovered. Will never be executed.

TRACE ABORT
A non-maskable interrupt has occurred.

ILLEGAL INSTRUCTION
An unknown CPU-instruction is detected in the interpreting mode.

END OF TASK <s>
The program terminates, where <s> may be the SVC error status.

12.9 EXAMPLE

The following example shows the use of the TRACE program.

-TRACE	Load and start the TRACE.
Trace Rx-yz	Signon from the program.
LOAD MYTASK	Load the task to be traced.
START MYTA	Start it. No parameters required.
-12.34.56 Paused	The TRACE-task pauses.
BC=0000 DE=0000 etc...	The current register set and the first
9000 SVC 2,9345 ..	instruction in the program is displayed.
SET MAIN=*	The RETURN-key was typed, and the current program location value is assigned to the symbol <main>.
TRACE MAIN+12A	Set up a single-step breakpoint at the relative address <12A> biased the symbol <main>.
PROCEED	Enter the trace mode at the current program location.
BC=0000 DE=0000 etc...	First the complete register set is shown,
9000 SVC 2,9234 ..	then the next instruction is displayed.
DE=003F	The single-key command 'D' was typed and
912A LI A,0D ..	the free-run mode was entered. Then the previous specified breakpoint was reached, and register changes as well as the new instruction is displayed.
END	The RETURN-key was typed to enter the command mode, where the END-command was entered.
-12.34.56 End-of-task 0	Program termination.
-CA	Remove the TRACE code.
-12.34.59 End-of-task 0	The TRACE task terminates.

FILE PATCH UTILITY (PATCH)

13.1 INTRODUCTION

The File Patch utility, PATCH, provides a facility for changing the code in both absolute and relocatable Task files. The new code is appended to a relocatable task, and inserted at their proper places in an absolute task.

13.2 SYSTEM REQUIREMENTS

The program requires:

-4 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

13.3 STARTING

The program is started by the command

PATCH(,O) TASKFILE

where <taskfile> is the name of the task to be changed, and <o> is used to select octal notation. Default notation is hexadecimal.

13.4 COMMANDS

The function of the program is controlled by commands entered in an interactive mode. Depending on the type of the taskfile, the program enters either the absolute, or the relocatable mode.

All values can be entered as expressions, which are evaluated from left to right on a strict 16-bit integer basis. Operands can only be numerics, either octal or hexadecimal depending on the start switch. Legal operations are +(PLUS) and -(MINUS).

When started, the program displays a menu of available commands:

ABSOLUTE MODE

EXamine ADDR,LENGTH
Display <length> bytes of the file starting at memory address <addr>.

MOdify ADDR,DATA(,DATA...)
Change the content of the file starting at memory address <addr> to absolute byte(s) with the value <data,...>.

Help
Displays the possible commands.

ENd
Terminates the program.

RELOCATABLE MODE

ABs ADDR,DATA(,DATA...)
Change the content starting at address <addr> to absolute byte(s) with value <data,...>.

RELAdr ADDR,RADDR(,RADDR...)
Change the content starting at address <addr> to relocatable address(es) with the value <raddr,...>.

RELBYte ADDR,DATA(,DATA...)
Change the content starting at address <addr> to relocatable byte(s) with the value <data,...>.

RELBY256 ADDR,DATA(,DATA...)
Change the content starting at address <addr> to relocatable byte(s) with the value <data,...) divided by 256.

HElp
Displays the possible commands.

ENd
Terminates the program.

13.5 MESSAGES

The program may output the following messages:

PATCH R.xyz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

BAD FILE TYPE
The file was not a task file.

COMMAND ERROR
When an unrecognized command is entered, or an illegal value.

ADDRESS OUT OF RANGE
Address specification is outside the limits of an absolute file.

END OF TASK <s>
The program terminates, and where <s> is the SVC error status.

13.6 EXAMPLE

This example shows how to modify a task file:

-PATCH FILENAME	Load and start the program, use hexadecimal notation.
Patch Rx.yz	Signon from the program, then the proper command menu is displayed.
ABS 1234,0,1	Puts out the absolute values ´0´ and ´1´ to the addresses ´1234´ and ´1235´.
RELA 89AB,1234	Puts out the relocatable address ´1234´ to the addresses ´89AB´ and ´89AC´.
END	Exit the program.
-12.34.56 End-of-task 0	Program terminates.

PROM PROGRAMMER (PROMPROG)

14.1 INTRODUCTION

The Prom-Program utility, PROMPROG, is used to transfer an absolute task file to EPROM-chips, which type could be either 2758, or 2716, or 2732.

The program is controlled through commands, and uses an internal buffer (programming buffer) to transfer the data to and from the EPROMs.

14.2 SYSTEM REQUIREMENTS

The program requires:

-10 KB of memory above the operating system, including buffer.

-a console device.

-a currently supported disc device.

-a prom programmer.

-a prom memory card.

14.3 STARTING

Connect the Prom-Programming card to any free I/O-slot. Mount the proper type-plug, which specifies the EPROM-type, on the Prom-Memory card. Select the base-address of the Prom-Memory card, and attach it to the Prom-Programmer.

When the program is started by the command

PROMPROG

the program displays a menu of available commands, the type-plug and the base address of the memory card. Then the program enters an interactive mode, and prompts on the console for commands.

14.4 COMMANDS

The function of the program is controlled by commands, that is entered in an interactive mode:

EXplain
    Displays the possible commands.

FIle FILENAME
Assign data input to <filename>.

Get PROMNR
Load the programming buffer with data for the EPROM with the number <promnr>. The <promnr> specified in this command, will be the default EPROM-number in all following commands, that has the <promnr> as an optional parameter.

NExt
Increments the default <promnr>, and loads the programming buffer with data, from the file, for next EPROM.

DUmp PROMNR
Transfers the content of the EPROM with number <promnr> to the programming buffer. This command is very useful when duplicating an already programmed EPROM.

INVert
To use inverted data or not.

TEst (PROMNR)
Test if the EPROM is erased or not.

PROgram (PROMNR)
Start program the EPROM with data from the programming buffer. When the EPROM is programmed, this command is followed by a verify phase.

VERify (PROMNR)
Verify the content of the EPROM with the content of the programming buffer. This is always done automatically after each ´PROGRAM´ command.

TYpe (PROMTYPE)
Sets the current EPROM-type to <promtype>. If the <promtype> is omitted, the possible EPROM-types are displayed.

RAdix 8/16
Selects base for numeric input and output. Default is 16.

ENd
Exit from the program.

14.5 QUESTIONS

Some commands responds with further questions, that has to be answered before any action takes place.

When the command ´INVERT´ is entered, the following question is raised:

DATA INVERTED OR NOT (Y/N)?
Reply ´Y´ or ´N´ if the data should be inverted or not.

When the optional parameter <promnr> is entered to the ´PROGRAM´ command, and the <promnr> is not equal to the default <promnr>, answer the following question:

ARE YOU SURE YOU WANT TO PROGRAM PROM s DATA INTO PROM d (Y/N)?
Reply ´Y´ or ´N´ depending on if this is true or not.

14.6 MESSAGES

The program may output the following messages:

PROM-PROGRAMMER R.xyz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

NO PROGRAMMER
Prom-programming card is missing.

PROGRAMMER NOT READY
The programmer doesn´t accept commands.

CARD SENSE ERROR
The memory card does not respond.

CARD ADDRESS ERROR
The base address, or the address range of the memory card is not matching the type of the EPROM.

PROM SIZE DOESN´T FIT PROGRAMMER
The memory card doesn´t respond to addresses within the size of the EPROM-type.

CARD ADDRESS RANGE: ssss - eeee
The PROM-card is starting at address <ssss> and ending at address <eeee>.

PROM SIZE bb KBITS
The size of each prom is <bb> Kilo Bits. Depends on the EPROM-type.

COMMAND DOES NOT EXIST, ENTER ´EXPALAIN´ FOR COMMAND MENU
When an unrecognized command is enetered.

INPUT FILE NOT ASSIGNED
Tries to load data prior to a ´FILE´ command.

INPUT FILE MUST BE IN ABS FORMAT
Input file is not an absolute task.

CHECKSUM ERROR
Wrong checksum of the task file.

FILE RANGE: ssss - eeee
The data in the task file starts at address <ssss> and ends at address address <eeee>.

ILLEGAL PROM NUMBER
Invalid PROM number, perhaps out of the allowed range 0-7.

BUFFER LOADED - CHECKSUM:cc
When the programming buffer is loaded either with the ´GET´, or the ´DUMP´ command. The checksum of the data is <cc>.

NO DATA
The programming buffer is empty.

PROM pp ADDRESS RANGE ssss - eeee
The EPROM number <pp> starts at address <ssss> and ends at address <eeee>.

PROGRAMMING...(*)(*)(...)
Programming of the EPROM has started. An asterisk is written for each modulo 256 bytes being programmed.

PROM pp VERIFICATION:
Verification of EPROM number <pp> has started.

OK
Verification without any mismatch.

ERROR IN ADDRESS: aaaa PROM=pp BUFFER=bb
Verify failed at address <aaaa>, data in the EPROM is <pp>, and the data in the programming buffer is <bb>.

MORE THAN 20 ERRORS
When there are more than 20 mismatches between the EPROM and the programming buffer, the verification stops.

PROM pp IS (NOT) ERASED, CHECKSUM=cc
EPROM NUMBER <pp> is/isn´t empty, current checksum is <cc>.

THAT TYPE IS NOT SUPPORTED
The type of the EPROM specified is not supported.

PROGRAMMING MODE NOW SET TO <PROMTYPE> TYPE
Informs of the current EPROM-type.

BAD RADIX
Radix missing or wrong radix.

END OF TASK 255
Program terminates.

14.7 EXAMPLE

This example shows how to program eight EPROMs with the same content:

```
-PROMPROG                    Load and start the program.
Prom-Program Rx.yz           Signon from the program.
TEST 0                       Check if the EPROMs are empty.
TEST 1
:
TEST 7
FILE TASKNAME                Assign file to be programmed.
GET 0                        Read data from the file.
PROGRAM 0                    Program and verify the EPROMs.
PROGRAM 1
:
PROGRAM 7
END                          Exit the program.
-12.34.56 End-of-task 255    Program terminates.
```

TEXT EDIT UTILITY (EDIT)

15.1 INTRODUCTION

The Text Edit Utility (EDIT) is a general purpose, ASCII-oriented, text editor to edit or create text interactively. The text may be any sequence of characters such as source programs, program data or documents.

Text is read sequentially from the input unit into an area of memory called the Edit-Buffer. This edit-buffer may be operated freely in any order, and the content of the edit-buffer may be written onto the output unit.

The EDIT takes advantage of any amount of additional memory for the edit-buffer, and any Operating System supported peripheral device can be used as input and output units.

The EDIT is edit-buffer oriented within the file, line oriented within the edit-buffer, and character oriented within a line.

This chapter gives only a breaf description of the EDIT, refer to DataBoard-4680 Edit Manual for futher information.

15.2 SYSTEM REQUIREMENTS

The program requires:

-13 KB of memory above the operating system, including 8 KB buffer.

-a console device.

-a currently supported text device.

15.3 STARTING

The program is started by the command:

```
EDIT(,,ADDMEM) INPUT(,OUTPUT)
```

The <addmem> will expand the edit-buffer.

The <input> is the name of the file or the device that contains, or shall contain, the text.

The optional <output> is the name of the file or the device where the result should be placed. If no <output> is specified, the EDIT uses a temporary file to hold the modified text.

15.4 COMMANDS

The function of the program is controlled by commands:

RE
Reads text from the input unit into the edit-buffer. The EDIT does not automatically reads the input unit. Editing an old text file starts by this command, which reads the first part of the input file into the edit-buffer. The edit-buffer is then open for all commands. If the text is larger than the edit-buffer available, the operator can successively edit through the input file in increments of the edit-buffer size. This is done by concurrent use of the RE command, editing, and the OR command.

PR (l1(-l2))
Displays the line(s) between, and including, <l1> and <l2>. The display may be aborted by simultaneously depressing the CTRL-key and the C-key.

WR
Writes the content of the edit-buffer to the new file. The content of the edit-buffer is maintained.

OR
Outputs the edit-buffer to the new file, clears the edit-buffer and the next part of the input-file is read into the edit-buffer. Line numbering will continue from the last line number in the preceeding.

IL (ln)
Lines of text are read from the terminal, and are inserted after either the last line in the edit-buffer, or the line <ln> specified in the command. Line numbers are assigned by the EDIT in increments of .01 following the current line. A number-sign (#) in the first column followed by a carriage return, terminates the insertion operation and returns to the EDIT command mode.

ln (s)
The line with the number <ln> is deleted if no string <s> is typed, else the string <s> will either replace an existing line with number <ln>, or be appended/inserted with the line number <ln>.

ED ln
This command is used to character-edit the line <ln>. After the carriage return key, the TAB-key (or CTRL/I) is used to display the content of the line. For each TAB-key one character is displayed. The CTRL/H key is used to delete a character, and to insert new characters, just enter them. This command is terminated by a carriage return.

DL l1(-l2)
Delete all lines between, and including, <l1> and <l2>.

KI
Deletes the content of the edit-buffer.

SV s
Searches for all occurences of the string <s>. Only one or no space is allowed to separate the command from the string. The string is a sequence of one or more ASCII characters that in the text are surrounded by any delimiter or non-alphabetic or numerical character.

CV s1(,s2)
Replaces every occurence of the string <s1>, with the string <s2>.

LC
Enables lower case input from a terminal with lower case set of keys.

UC
Forces lower case input from a terminal to be converted to upper case. Concerns terminals with UC/LC facility.

NU
Renumber the edit-buffer.

BT t1,t2,t3
This command is used to set the tabulation columns. Default is 11, 17, 35. The tabulation function is generated by the TAB-key, or CTRL/I.

AB
The session is aborted, and neither changes nor a new file will be done.

EN
Finishes the EDIT session. The content of the edit-buffer, if any, is written to the output file, and if more text is present in the input file, this text is copied to the output file. If the output file is a temporary file, it is renamed to the name of the input file. The input file, if any, is renamed by replacing/appending the last character to a '&'.

15.5 MESSAGES

The EDIT outputs the following messages:

EDIT Rx.yz
Signon from the program, where the revision level is <x>, and the update level is <yz>.

BAD PARAMETERS
Parameters missing, or syntax error in file descriptors.

ASSIGN ERROR
Failed to assign the file(s).

NO OUTPUT FILE
Output file missing.

TOO NAMY EDIT TEMP FILES
Failed to create a temporary file.

ERROR s
    Output error <s>.

BAD COMMAND
    Unknown command entered.

LINE TOO LONG
    The text line exceeds eighty characters.

CAN´T FINT IT
    Unable to find the string requested in a SV command.

SYNTAX ERROR
    Parameter(s) missing, or invalid.

END OF MEMORY
    End of edit-buffer at insert or append.

END OF TASK s
    Task termination, where <s> is the SVC error status.

15.6 EXAMPLE

This example shows how to edit an existing file:

```
-EDIT FILE                Load and start the program.
Edit Rx.yz                Signon from the program.
>RE                       Load the edit-buffer.
>IL 2                     Insert new text after line number two.
2.01#SOME NEW TEXT        Type in the new text.
2.02##                    Terminate the insertion.
>EN                       Exit the program.
-12.34.56 End-of-task 0   Program terminates.
```

MACRO ASSEMBLER (ASMZ)

16.1 INTRODUCTION

The MACRO ASSEMBLER (ASMZ) provides the capability to assemble a program for both the Z-80 and the 8080 Processors. The ASMZ produces relocatable object code and has a full complement of user features, including macro handling, conditional assembly, page control, arithmetic expressions, and support of common. It contains over 40 pseudo-ops. Refer to DataBoard-4680 Assembler Manual for more detailed information.

The ASSEMBLER consists of a main program named ASMZ, and an overlay named ASMZOVL. The ASMZ compiles source statements to a temporary work file, and the ASMZOVL updates an object library with the content in the temporary work file.

16.2 SYSTEM REQUIREMENTS

The program requires:

-21 KB of memory above the operating system, including buffer.

-currently supported devices.

16.3 STARTING

The program is started by the command

ASMZ(,(E)(N)(I)(C)(,ADDMEM)) SOURCE(,(OBJECT),(COPY)(,LIST))

The switch <e> is used to generate an error listing only.

<n> disables the listing of the macro expansion.

<i> enables the listing of the conditional assembly.

<c> enables the 8080 target code checking.

The <addmem> is the additional size in bytes, that should be added to the ASMZ to hold the symbol table. A default table sufficient for 100 symbols is normally contained within the ASMZ.

The parameter <source> is the name of the source file to be assembled.

<object> is the the name of the object library, created if not existing, to whom the assembled module shall be appended or replaced.

<copy> is the name of the unit, from which source statements shall be included.

<list> is the name of the file or device that will receive the listing.

16.4 MESSAGES

The program may output the following messages:

MACRO ASSEMBLER Rx.yz
Signon by the program, where the revision level is <x>, and the update level is <yz>.

BAD START PARAMETERS
Syntax error in start parameters.

SOURCE ASSIGN ERROR
Failed to assign the source input.

COPY FILE ASSIGN ERROR
Failed to assign the copy file.

LIST ASSIGN ERROR
Failed to assign the list unit.

TEMP FILE ASSIGN ERROR
Failed to create and/or assign a temporary work file.

ERROR s ON SOURCE LU
Input error <s> on the source unit.

WRITE ERROR s ON OBJECT LU
Output error <s> on the temporary work file.

END OF MEMORY
End of memory for the label list.

ERRORS: e WARNINGS: w
There where <e> errors and <w> warnings in the program.

END OF ASSEMBLY
The assembly phase terminates.

ASMZOVL MISSING
The assembler overlay is missing.

OBJECT UPDATER Rx.yz
Signon by the assembler overlay, where the revision level is <x>, and the update lavel is <yz>.

TEMP FILE IS EMPTY
There is no program in the temporary work file.

LIBRARY CREATED
A new object library is created.

PROGRAM: name REPLACED/APPENDED
The program named <name> is replaced or appended to the object library.

END OF TASK <s>
The program terminates, and where <s> is the SVC error status.

16.5 EXAMPLE

This example shows how to assemble a file name SOURCE:

```
-ASMZ SOURCE                Load and start the program.
MACRO ASSEMBLER Rx.yz       Signon from the program.
ERRORS: 0  WARNINGS: 0      No errors and no warnings.
END OF ASSEMBLY             Assembly phase terminates.
-12.34.56 End-of-task 0     Program terminates.
```

FORTRAN-77S (FORT77)

19.1 INTRODUCTION

FORTRAN-77S is a high level language processor intended to support the programming requirements of not only both scientific and process-control applications, but also when to use file manipulation. The compiler produces relocatable object code. It is a two-pass high level language processor that supports the ANSI standard FORTRAN 77 (X3.9-1978) subset, with extension of some parts of full language. PROMable code may be generated by the Task Establisher, ESTAB.

The FORTRAN system consists of the compiler, named FORT77, and two run-time libraries named FORTRTLS and FORTRTLM.

This chapter is only a breaf description and more detailed information will be found in DataBoard-4680 Fortran Manual.

19.2 SYSTEM REQUIREMENTS

The program requires:

-30 KB of memory above the operating system, including buffer.

-currently supported devices.

19.3 STARTING

The program is started by the command

FORT77(,(D)(X)(,ADDMEM)) SOURCE(,(OBJECT)(,LIST))

The switches <d> and <x> are used to compile statements that has either a ´D´ or a ´X´ in its first column.

The <addmem> is used to expand the symbol table.

The parameter <source> is the name of the source file to be compiled.

<object> is the the name of the object library, created if not existing, to whom the compiled module shall be appended or replaced.

<list> is the name of the file or device that will receive the listing.

19.4 OPTIONS

Compilation options may be inbedded in the source code and should be preceded by a $ sign. Following is a list of available options that may be used during the compilation phase:

BATCH
: Enables batch compilations of modules separated by END statement.

BEND
: Terminates batch compilation.

LIST
: Enables listing of source statements.

NLIST
: Disables listing of source statements.

NODBG
: Supresses listing of location counter.

EJECT
: Starts a new page.

LCNT n
: Changes paper hight to value <n>. Must be greater than 14, and will be activated from the next page.

TITLE text
: Moves up to 60 characters of <text> to list heading.

PREC n
: Changes integer constant calculation precision to <n> bytes, where <n> can be 1, 2 or 4. The default precision is 2.

19.5 LINKING

When linking the final task, some unique declarations must be entered to the ESTAB program:

```
PLCORD 0,1,2,3   The order of the involved PLC´s.
                 PLC 0 contains instruction, PROMable.
                 PLC 1 contains data, PROMable.
                 PLC 2 contains variables, not PROMable.
                 PLC 3 is used for scratches, not PROMable.
EQU 0,LU.CON     Logical unit for the console.
EQU 5,LU.PR      Logical unit for the printer.
LIB FORTRTLS     Runtime library select file.
LIB FORTRTLM     Runtime library module file.
```

19.6 MESSAGES

The program may output the following messages:

FORT77S Rx.yz
: Signon by the program, where the revision level is <x>, and the update level is <yz>.

BAD START PARAMETERS
: Missing or syntax error in start parameters.

CAN´T ASSIGN SOURCE/OBJECT/LIST
: Failed to assign the source/object/list units.

OUT OF MEMORY
: End of memory for the symbol table, restart the compiler with additional memory.

ERROR s ON SOURCE LU
: Read error <s> on the source input.

I/O ERROR s ON SCRATCH LU n
: Write error <s> on the scratch logical unit <n>

COMPILATION ABORTED
: The compiler terminates abnormal before it has reached end of file in the source file.

END OF COMPILATION
: The compilation phase terminates normal.

END OF TASK <s>
: The program terminates, and where <s> is the SVC error status.

19.7 EXAMPLE

This example shows how to compile a file named TEST.

```
-FORT77 TEST                Load and start the program.
FORT77 Rx.yz                Signon from the program.
NO COPILATION ERRORS        The compilation was successful.
END OF COMPILATION          The compiler phase terminates.
-12.34.56 End-of-task 0     Program terminates.
```

APPENDIX A

COMMAND SUMMARY

ALlocate(,I) fd(, (lrecl) (, (size)(/blk)))
  Allocates indexed file <fd> with
    <lrecl> = record length default = 0, variable
    <size> = file size in sectors default = as initiated
    <blk> = data block size in sectors, default = as initiated

ALloacte,C fd,size
  Allocates contigouos files <fd> with
    <size> = file size in sectors

BIas address
  Sets BIAS to <address>.

CAncel (tid)
  Cancels task named <tid>.

CLOse fd
  Closes device <fd> to the Operating System.

COntinue (tid)
  Continues task named <tid>, if paused.

DELete fd(,fd...)
  Deletes specified file(s).

DEVices (fd)
  Displays on <fd> the names of the devices and direct access volumes in the system.

EXamine address(,n)
  Displays the contents of the <n> locations, starting with <address>+BIAS.
    <n> defaults to 2

Lib ((voln:)(directory),(wildcard),(list))
  Displays the content of the volume <voln> and the directory file <directory>, file names is specified by <wildcard>, and the output is directed to <list>.

LOad fd(, (tid)(,size))
Loads task from <fd>, names it <tid> and adds memory <size>.

MOdify address,data(,data...)
Changes location specified by <address>+BIAS to <data>.

OPEn(,(N)(P)) fd
Opens device <fd> on-line to the Operating System. Protect <P> specifies device is read-only, and Non-filestructured <N> specifies no directory present.

OPTion,opt(opt...) tid
Sets task named <tid> to <opt>. Possible <opt> are Abortable, Protected, Resident, Nonresident.

PAuse (tid)
Pauses task named <tid>.

POSit fd,cmd(,n)
Position <fd> according to <cmd> times <n>, where <cmd> is BF, BR, FF, FR, REW, RW or WF.

PRIority tid,n
Sets priority of <tid> to <n>.

RAdix 8/16
Set radix to octal (8) or hexadecimal (16).

RENname oldfd,newfd
Renames file <oldfd> to name <newfd>.

RUn(,switches) fd(,parameter)
Load and start the program on the file <fd>. Pass <switches> and transfer <parameters> to the task.

SLice (n)
Set and/or display the time slicing constant in milliseconds.
<n>=0, switches off time slicing.

SPace (voln)
Displays the space available on <voln>.

STart(,switches) tid(,paramaters)
Start task with taskid <tid> at ordinary start address. Pass <switches> to the tasks registers, and transfer <parameters> to the tasks stack.

TAsk(,F) (fd)
  Display information about the present task(s) on <fd>.

TIme (YYYY-MM-DD, HH.MM.SS)
  Set or display the date and time.

Volume (voln)
  Set or display the name of the system volume.

APPENDIX B

CRASH LAYOUT

```
II=ii CC=cc IL=il CS=cs TP=tp TN=tn TC=ttcb SS=sspp    (SPT)
PC=ppcc SP=sspp  F=flag  A=aa BC=bbcc DE=ddee HL=hhll  (Primary)
 Y=yyyy  X=xxxx  F=flag  A=aa BC=bbcc DE=ddee HL=hhll  (Secondary)
SS
addr aabbccdd .... .... .... .... .... .... vvxxyyzz  (System stack)
aadr aabbccdd .... .... .... .... .... .... vvxxyyzz
SP
addr aabbccdd .... .... .... .... .... .... vvxxyyzz  (Current stack)
addr aabbccdd .... .... .... .... .... .... vvxxyyzz
```

System pointer table (SPT):

II = illegal interrupt counter.
CC = crash code.
IL = interrupt level, system mode if less than 80.
CS = card selection code.
TP = task priority.
TN = task number.
TC = task control block.
SS = system stack-pointer.

Primary register set:

PC = program location counter.
SP = program stack-pointer.
 F = condition code.
 A = A-register.
BC = BC-register pair.
DE = DE-register pair.
HL = HL-register pair.

Secondary register set:

 Y = Y-index register.
 X = X-index register
 F = condition code.
 A = A-register.
BC = BC-register pair.
DE = DE-register pair.
HL = HL-regsiter pair.

APPENDIX B (Continued)

CRASH CODES

NR	SYMBOLIC	DESCRIPTION
1	CC.COLD	SVC-8 failed in cold start.
2	CC.USER	User crash entry.
3	CC.ZERO	Entry trough address zero.
4	CC.NMI	Too many non-maskable interrupts.
5	CC.VINT	Too many un-exspected vector interrupts.
6	CC.INTER	Too many illegal interrupts in 4680 I/O-structure.
7	CC.TIME	Real-time service never done, loops on levels =< 10.
8	CC.SETSL	Attempt to set up a non-existing level.
9	CC.TRGSL	Attempt to trigg a non-existing level.
A	CC.ICBIL	Attempt to trigg an ´ICB´ with a non-existing level.
B	CC.SINT	Attempt to execute on a non-existing level.
C	CC.ADDIL	Attempt to add a ´DCB´ on a non-existing level.
D	CC.RMVIL	Attempt to remove a ´DCB´ from a non-existing level.
E	CC.RMVIQ	Failed to remove a ´DCB´ from an interrupt chain.
F	CC.ADDRQ	Attempt to add a ´TCB´ twice to Ready-Q.
10	CC.RMVRQ	Failed to remove a ´TCB´ from Ready-Q.
11	CC.RMVBQ	Failed to remove a ´TCB´ from Buffer-Q.
12	CC.ADDSQ	Attempt to add a ´RCB´ twice to System-Q.
13	CC.RMVSQ	Failed to remove a ´RCB´ from System-Q.
14	CC.TASK	Attempt to find a non-generated task-number.
15	CC.STOP	Attempt to stop a task in System Mode.
16	CC.DESC	Attempt to connect at decendent in a tree.
17	CC.PROP	Attempt to propagate priority on a non-existing task.
18	CC.TTDEV	Attempt to trigg a non-existing task-device.
19	CC.DISP	Attempt to dispatch a non-existing task.
1A	CC.DISC	Attempt to disconnect a non-existing task.
1B	CC.RMVTQ	Failed to remove a node from ´SPT.MLNK´.
1C	CC.NODE	Attempt to release a non-existing node.
1D	CC.QUEUE	A queue has become a circular list.
1E	CC.RMTQ	Failed to remove a ´RMT´ from symbolic chain.
1F	CC.RRTQ	Failed to remove a ´RRT´ from numeric chain.
20	CC.TCAN	Failed to cancel a task.
21	CC.AMEM	Attempt to allocate memory twice.
22	CC.RMEM	Attempt to release memory twice.

APPENDIX B (Continued)

CRASH CONDITIONS

NR	CONDITION
1	A := Return status, Y -> SVC-blk, HL -> next in cold start table.
2	Specified by user.
3	SP -> last item pushed on stack.
4	SP -> previous A.
5	SP -> previous A.
6	A := current system level, X -> previous DCB in chain.
7	SP -> BC, AF, HL, PC.
8	C := non-existing system level.
9	L := non-existing system level.
A	A := non-existing system level in an ICB.
B	A := non-existing system level.
C	A := non-existing system level in an ICB, X -> DCB.
D	A := non-existing system level in an ICB, X -> DCB.
E	X -> DCB, DE -> ICB.IQLK, HL -> root of chain.
F	X -> TCB.
10	X -> TCB, DE -> TCB.RQLK, HL -> SPT.RQUE.
11	X -> TCB, DE -> TCB.RQLK, HL -> SPT.BUFQ.
12	X -> RCB, DE -> RCB.SQLK.
13	X -> RCB.
14	B := task number.
15	SP -> AF1, BC1, DE1, HL1, CSIL, Y, X, AF, BC, DE, HL, PC.
16	X -> RCB, Y -> SVC-blk, B := SVC-type, C := resource number.
17	B := task number, (SP) := RCB, DE -> xxx.CPRI.
18	B := task number, DE -> node, (SP) := DCB.
19	B := task number, (SP) := RCB.
1A	B := task number, SP -> Y, X, PC.
1B	DE -> node, HL -> SPT.MLNK.
1C	DE := invalid node address.
1D	B := 0 !
1E	C := resource number, DE -> RMT.
1F	C := resource number, DE -> RRT.
20	X -> TCB.

APPENDIX C

ERROR CODES

COMMON ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
000	00	0	S0S.OK	No error.
001	01	1	S0S.EON	End of nodes.
002	02	2	S0S.IFC	Invalid function code.
003	03	3	S0S.PRO	Can't connect at unconditional proceed.
004	04	4	S0S.OFFL	Off line.
005	05	5	S0S.PRES	Not present in this system.
006	06	6	S0S.NYET	Not yet implemented function.
007	07	7	S0S.CAN	Request is cancelled.
010	08	8	S0S.SVC	Invalid SVC function.

SVC-1 I/O ERROR CODES

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
012	0A	10	S1S.LU	Illegal LU, LU not assigned.
013	0B	11	S1S.AM	Invalid access modes.
014	0C	12	S1S.TOUT	Time-out.
015	0D	13	S1S.DWN	Device down.
016	0E	14	S1S.EOF	End-of-file.
017	0F	15	S1S.EOM	End-of-media.
020	10	16	S1S.RER	Recoverable error.
021	11	17	S1S.UNR	Unrecoverable error.
022	12	18	S1S.RND	Invalid random address.
023	13	19	S1S.NRND	Non-exsistent random address.

SVC-2 SUBFUNCTION ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
024	14	20	S2S.ISB	Illegal sub-function number.

SVC-3 TIMER ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
036	1E	30	S3S.PAR	Invalid timer parameter.

SVC-4 TASK DEVICE ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
050	28	40	S4S.ASGN	Not assigned.
051	29	41	S4S.TYPE	Invalid device type.

SVC-5 LOADER ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
062	32	50	S5S.TID	Illegal task-id.
063	33	51	S5S.PRES	Task present.
064	34	52	S5S.PRIO	Illegal priority.
065	35	53	S5S.OPT	Illegal option.
066	36	54	S5S.CODE	Illegal code/item at load.
067	37	55	S5S.SIZE	Overlay don't fit.

SVC-6 TASK ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
074	3C	60	S6S.TID	Illegal task-id.
075	3D	61	S6S.PRES	Task present.
076	3E	62	S6S.PRIO	Illegal priority.
077	3F	63	S6S.OPT	Illegal option.
100	40	64	S6S.EQUE	Event queue disabled.
101	41	65	S6S.STAT	Invalid task status.

SVC-7 FILE ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
106	46	70	S7S.ASGN	Assignment error, double assign.
107	47	71	S7S.AM	Illegal access modes.
110	48	72	S7S.SIZE	Size error.
111	49	73	S7S.TYPE	Type error.
112	4A	74	S7S.FD	Illegal file descriptor.
113	4B	75	S7S.NAME	Name error.
114	4C	76	S7S.KEY	Invalid key.
115	4D	77	S7S.FEX	File exist error.

SVC-8 RESOURCE ERRORS

OCT	HEX	DEC	SYMBOLIC	ERROR TEXT
120	50	80	S8S.ID	Illegal resource-id.
121	51	81	S8S.CLAS	Invalid resource class.
122	52	82	S8S.PRES	Resource already present.
123	53	83	S8S.PRNT	Parent not present.
124	54	84	S8S.DUAL	Dual DCB not present.
125	55	85	S8S.RCB	Invalid RCB-type.
126	56	86	S8S.EOM	End-of-memory.