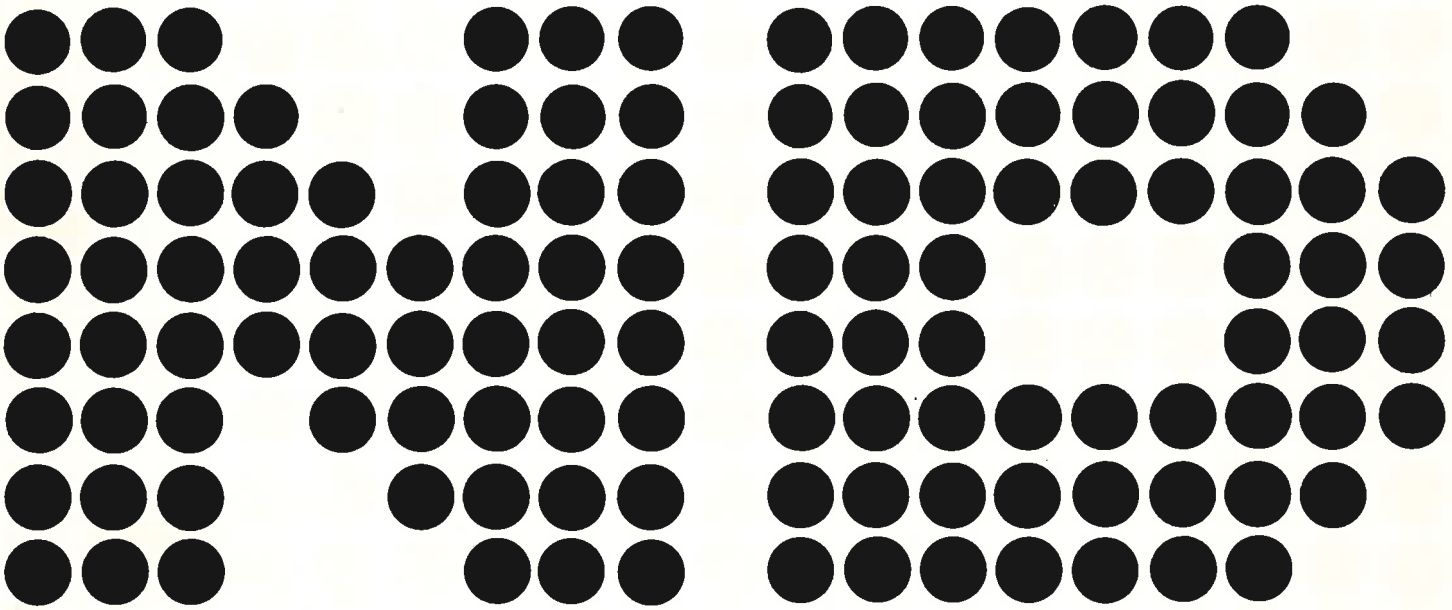
MACM

Mac Mass Storage Assembler

NORSK DATA A.S



MACM

Mac Mass Storage Assembler

REVISION RECORD

Revision	Notes
11/74	Total revision superceding all previous versions

TABLE OF CONTENTS

		Page
1	PURPOSE OF MACM	1-1
2	DRUM LAYOUT	2-1
3	MACM COMMANDS	3-1
3.1	)GJEM	3-1
3.2	)HENT	3-1
3.3	)SAM	3-1
3.4	)CMOVE	3-1
3.5	)BPUND	3-1
3.6	)CLOAD	3-1
3.7	)FIX	3-2
3.8	)SYSDF	3-2
3.9	)ULIST	3-2
3.10	)LINE	3-3
3.11	)XPUN	3-3
3.12	)9READ	3-3
3.13	)9BYTT	3-4
3.14	)9CTOM	3-4
4	ASSEMBLER OPTIONS USED WITH MACM	4-1
5	DEBUGGING AND RUNNING USER PROGRAMS	5-1
5.1	Start User Program	5-1
5.2	Return to MACM	5-1
5.3	Breakpoint	5-1
6	REBOOTABLE TAPE	6-1
6.1	Loading	6-1
6.2	Format	6-1
7	ADDRESSING THE DRUM (Regards DRUM only)	7-1
8	GENERAL	8-1
9	SPECIAL PURPOSE COMMANDS FOR GENERATING SINTRAN III	9-1

1 PURPOSE OF MACM

MACM is a modified standard MAC assembler. The main difference is its ability to assemble programs out on a mass storage device (drum) in a core image format. Later appearance of "drum" must be considered as disc if this is the actual mass storage.

When MACM is working, it has complete control of CPU and external devices.

MACM has also the capability to swap itself with the core image on mass storage and start in a specified location. Used together with the two program tapes, User Break Point (UBP) and Core to MACM transfer (CTOM), the user may freely change between the MACM assembler and his own program much the same way as with an ordinary MAC assembler. However, the problem of protecting MACM from the user program is non-existant and all core is available to the user.

As an aid to debugging MACM has been equipped with commands to save ( )GJEM) and restore ( )HENT) the current core image to and from a save area on the drum. The saved area may be compared with the core image area word by word by means of the compare command (a<b;)SAM). The core image area may be loaded from core using CTOM after having run the user program.

The contents of a coreload may be moved to another coreload ( )CMOVE).

A rebootable tape may be punched out using the core image as source ( )BPUND).

When this tape is placed in the reader and the LOAD-button pushed, it will automatically be read into core and written back on the drum.

An additional mode, "system definition mode", may be used when assembling application programs.

This mode ensures that only those system-symbol-definitions which are referred to will be taken care of, others are ignored, when reading in a list ( )LIST) of definitions from a system or main program.

In order to use MACM to link programs separately assembled, the undefined reference list may be punched out ( )ULIST).

2 DRUM LAYOUT

MACM assembles programs in a core image on the drum. This core image may physically be one or more areas of the drum storage, however, it's combined size will always equal that of core.

A core image consits of:

1 Core resident core image excluding the coreload area,

and

2 The actual coreload as specified by the)CLOAD command (see page 3-1).

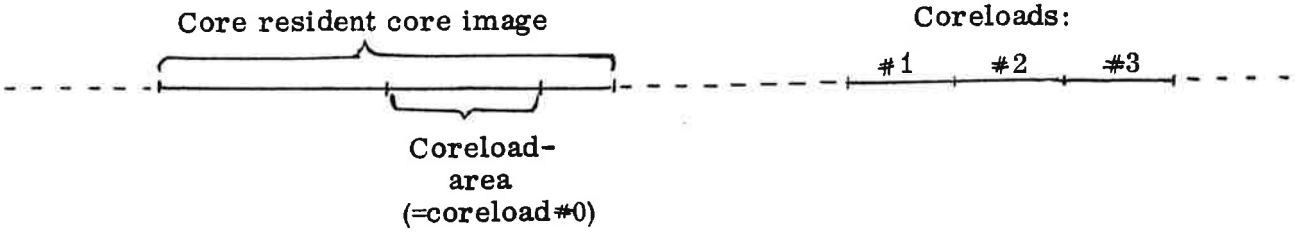


Figure 1. Layout of drum storage.

In SINTRAN Monitor coreloads are numbered from 1 (one) and upwards.

In MACM coreload number 0 (zero) denotes the coreload area of the core resident core image. Thus, if coreload number 0 is chosen, MACM will work on a contiguous core image area on the drum.

3 MACM COMMANDS

3.1)GJEM

copies the complete current core image to a save area on the drum. Current coreload number is saved for later use by)HENT.

3.2)HENT

restores the core image from the save area, using the coreload number saved by)GJEM.

3.3)SAM

compares word by word the core image area and the save area. Any differences are printed on the Teletype. The)SAM command is used with lower and upper limits for the comparison:

A < B

)SAM

Words from address A to address B both inclusive are compared.

3.4)CMOVE N1 N2

moves (copies) the contents of coreload number N1 into coreload number N2.

3.5)BPUND

punches a rebootable tape (see page 6-1) of the area defined by lower and upper limits (A<B) in the current core image.

3.6)CLOAD N1

defines the current coreload number, N1, to be used by MACM henceforth.

The number N1 is checked to see if it is within allowable limits (≥ 0 and $\leq$CLM), and further, if there are undefined symbols in the table the warning message "UNDEF SYMBOLS" is printed. This is due to the fact that MACM does not know in which coreload undefined symbols are referred. - Thus it is the responsibility of the user to have the proper coreload number set when symbols which have been referred are defined.

If the symbol, C has been entered in the Local Symbol Table, it will always have a value equal to current coreload number. To have the current coreload number printed out ",c" is typed on the Teletype and MACM will print out the number.

3.7)FIX

makes all symbols in the Local Symbol Table permanent. That is, they will not be deleted by any)CLEAR command given later on.

This command is intended to be used to make global (or system) variables permanent when assembling a system from parts.

3.8)SYSDF

puts MACM in a System Definition Mode. This mode is reset by)LINE (20).

While in System Definition Mode only those symbols referred to in the Undefined Symbol Table will be defined when inputting an equal sign definition to MACM. All other equal sign definitions are ignored.

The purpose of this mode is to avoid filling up the tables with unnecessary symbols.

3.9)ULIST

is a punch command making it possible to link several separately assembled, but interrelated programs using the assembler.

)ULIST outputs the Undefined Reference Table in symbolic code with the following format:

<octal address>/↑<undefined symbol name> % previous contents in this location

Proposed use:

Each program part is separately assembled and)LIST,)ULIST and)BPUND tapes are produced for each part. Finally the different parts are linked together by the following three step procedure:

1. Load all binary tapes
2. Input all)ULIST tapes to MACM
3. Input all)LIST tapes to MACM

The System Definition Mode ( )SYSDF) may successfully be used in step 3.

3.10)LINE or @

In addition to the previous definition of this command, "return input control to the Teletype", the following will also be effectuated:

1. Reset System Definition Mode (see)SYSDF, above)

and

2. Update core image on mass storage by emptying the mass storage block buffer in MACM.

3.11)XPUN

This command is included in order to punch out a standard binary tape of MACM itself.

)XPUN may punch out several disjoint areas of core:

```
a < b
)XPUN↵

c < d
)XPUN↵

e < d
)XPUN↵
```

Finally the lower and upper limits are zeroed and the)XPUN command is followed by the symbolic start address:

```
0 < 0
)XPUN MAC↵
```

3.12)9READ

reads binary tape produced by the)BPUN command into the current core image. The octal part is ignored, i.e. the command searches for the exclamation mark.

The contents of the binary tape are placed on the core image between the limits specified for)BPUN.

Note that the command takes input from the device specified for object output ! Remember to reset this device number immediately after the)9READ command.

3.13)9BYTT < Ten Symbols separated by Space >

This command makes it possible to change the "basic parameters" of a MACM system. This means, for example, that the same binary version of MACM may be used for drum as well as disc. The ten symbolic parameters for)9BYTT must be previously defined. The meaning of the parameters is explained below:

MSTYP = Mass storage type. Drum = 0, NCR-disc = 1, CDC-disc = 2, Large disc = 3.
DEVNO = Primary mass storage device number
CORAD = Start address of coreload in core
LONG = Length of coreload in words
CLM = Upper limit for coreload numbers (inclusive)
BLST = "Mass storage address" of coreload number one
DRES = "Mass storage address" of core resident core image
CRMAX = End of core address (37777 for 16K core)
MACAD = "Mass storage address" of area where MACM is saved
DASA = "Mass storage address" of "GJEM/HENT" area.

After the symbols have been given the desired values, type the command:

)9BYTT ␣MSTYP ␣DEVNO ␣CORAD ␣LONG ␣CLM ␣BLST
␣DRES ␣CRMAX ␣MACAD ␣DASA␣

MACM now writes CR-LF indicating that the command has been executed. If a symbol in the parameter string is not defined, the error message:

FABS NOT FOUND

is printed. This means fixed absolute symbol does not exist in MACM's symbol table. Restart with)9BYTT.

The symbol names may of course be anything, but the order of the parameters is essential (as described).

3.14)9CTOM

is connected to the object stream. It produces an octal tape described in Section 5.2. It is important to remember that some parameters given to)9BYTT are used. Thus a "CTOM-tape" must always correspond to the (latest))9BYTT command.

4 ASSEMBLER OPTIONS USED WITH MACM

Several of the options to MAC may, with some changes made, be used with MACM.

The main modification regards accessing assembled code. In MACM this must be done using a subroutine (DGET) while in MAC this can be done directly.

All standard options are available.

5 DEBUGGING AND RUNNING USER PROGRAMS

5.1 Start User Program

Transferring control from MACM to a user program is done the usual way by writing a start address followed by the exclamation mark, <Defined expression>!. This will cause MACM itself to be saved on the drum in a MACM save area, the current user core image is read into core and control is transferred to the specified location.

NOTE: The upper 28 locations of the user core image should not be used as these are used by the transfer routine.

Using start address 0 (zero) will not cause a transfer to location zero, but rather force a JMP * to be executed in the transfer program when swapping is finished.

5.2 Return to MACM

When the user has tried executing his program, he may want to return to MACM either to make corrections in his current program or for reload of core of a fresh program.

This may be done with a 2-part program tape in octal format read by the hardware assembler called CTOM (Core TO MACM).

The first part is used if the user wants to save his current core status, i.e. core is written on the core resident core image area of the drum.

The second part causes MACM to be loaded into core from its save area and started.

These two parts of the CTOM tape are separated by some blank tape so as the user may only use the second part if he so desires. However, this will cause his current core status to be lost.

5.3 Breakpoint

In order to make use of the breakpoint function a User Break Point tape (UBP) must be assembled together with the user program. UBP will cause an automatic core swapping and start of MACM whenever a breakpoint is reached via location 0 and 2.

The ordinary breakpoint commands in MAC are applicable:

<address expression >.	Set breakpoint at specified location.
1!	Execute one instruction by advancing the breakpoint.
!	Continue from last breakpoint (may even be used without changing last breakpoint location).
)RBP	Forget all about breakpoints
0.	Set no breakpoint
<address expression>!	Start user program in specified location.

NOTE: Breakpoints may only be used with the core resident core image area (no breakpoint may be set on coreloads 1, 2.....).

6 REBOOTABLE TAPE

A rebootable tape is defined as binary tape that may be autoloaded onto the drum. Rebootable tape is produced with the)BPUND command in MACM (see page 3-1).

6.1 Loading

A rebootable tape is loaded as follows: Deposit the primary mass storage device number into core address zero. After the tape has been placed in the proper reader, the LOAD-button is pushed and part of the tape is read into core and written on the drum. This is repeated until finished if no errors occur during loading. When the loading process stops, the computer has executed a WAIT-instruction. By examining the lower 8 bits of the IR-register the user may determine whether the load was successful or not:

WAIT 7	Tape loaded without error
WAIT 77	Checksum error on last binary block read A-reg. = checksum discrepancy
WAIT 376	Drum transfer error T-reg. = 0 or 1, READ/WRITE respectively D-reg. = drum block address

6.2 Format

A rebootable tape contains the following parts:

a. Standard Binary Load program (SBL) containing the start address of part b

b. Load Control Program (LCP)

c. Drum Transfer Routine (DRUM)

d. Drum Block Parameters (core address, drum block address, 0, last block flag)

e. Drum Block Contents (value of those words of the block to be loaded)

f. Parts d. and e. are repeated for each drum block affected

Part a. is in octal format, while each of the other parts are in standard MAC binary format.

The standard MAC binary format looks like:

1. and 2. tape frame:	Core address of first word
3. and 4. tape frame:	Number of 16-bits words

5. and 6. tape frame: - - - (2N+3). and (2N+4). tape frame:	Contents of N 16 bits words
(2N+5). and (2N+6). tape frame:	Checksum (= negative of the sum of the N contents)

(2 tape frames equal one 16 bits word).

The loading process proceeds as follows:

1. The Standard Binary Load program is read by the Octal Hardware loader and is started automatically.
2. The Load Control program is read by the Standard Binary Load program and started.
3. The Load Control program (LCP) calls the Standard Binary Load program (SBL) as a subroutine to read into core the Drum Transfer routine (DRUM).
4. LCP calls SBL to read Drum Block Parameters.
5. LCP calls DRUM to transfer the pertinent block from the drum to core. (DRUM loads the actual device number from core address zero.)
6. LCP calls SBL to read Drum Block contents.
7. LCP calls DRUM to write the modified drum block onto the drum.
8. LCP tests the fourth parameter in the Drum Block Parameters to see if this was last block. If not the sequence is repeated from 4.

7 ADDRESSING THE DRUM (This chapter regards DRUM only.)

In order to determine the drum address assembly time parameters for MACM, some knowledge of drum addressing is required.

The basic hardware drum addressing scheme works by assigning a distinct block address for each block of 64-words. The hardware block address equals (track number * 32 + block number). On each track there are 32 blocks numbered from 0 to 31. The interfacing hardware equipment makes it possible to transfer one or more blocks within the same track with one activation call from the program.

Example: If track 1 and block 29 (hardware block address 61) is addressed and 5 blocks are to be transferred, the following blocks on track 1 will be transferred: 29, 30, 31, 0 and 1.

Programs are usually not interested in this sequence, but rather blocks 29, 30 and 31 on track 1 and blocks 0 and 1 on track 2. The basic DRUM transfer routine should therefore know when the end of a track is reached and start a new transfer on the next higher track. The DRUM routine may consequently divide a programmed call into several activations of the drum hardware.

There is only one disadvantage left to be taken care of: When the end of a track is reached, there is not time enough to set up another call before the drum has rotated past the first block on next track. A delay of one rotation (10 ms) is therefore introduced.

This is avoided in the DRUM routine by rotating the block address one block for each track. Thus only a delay of $1/32 \cdot 10$ ms is introduced. This means that the DRUM routine must translate the programmers drum address into another hardware drum address. The algorithm is:

$$\text{Hardware drum address} = 32 \cdot S + [B + (S) \bmod 32] \bmod 32$$

where $S = \text{ENTIER}(\text{Drum address}/32)$

$B = \text{Drum address} - 32 \cdot S$

Consequently, please note the two differing terms for drum addressing: "Drum address" and "Hardware drum address".

Usually the "drum address" is used, however, in certain cases the corresponding "hardware drum address" must be used (specialized short drum transfer routines, CTOM etc.).

However in the SINTRAN Monitor system there is a third drum addressing concept, the "software drum address". This is due to the fact that SINTRAN works with units of information on mass storage called software blocks. A software block is usually 256 words long and thus equals 4 drum blocks.

The relationship to the "drum address" described above in this case is:

"software address" = $\frac{1}{4}$. drum address

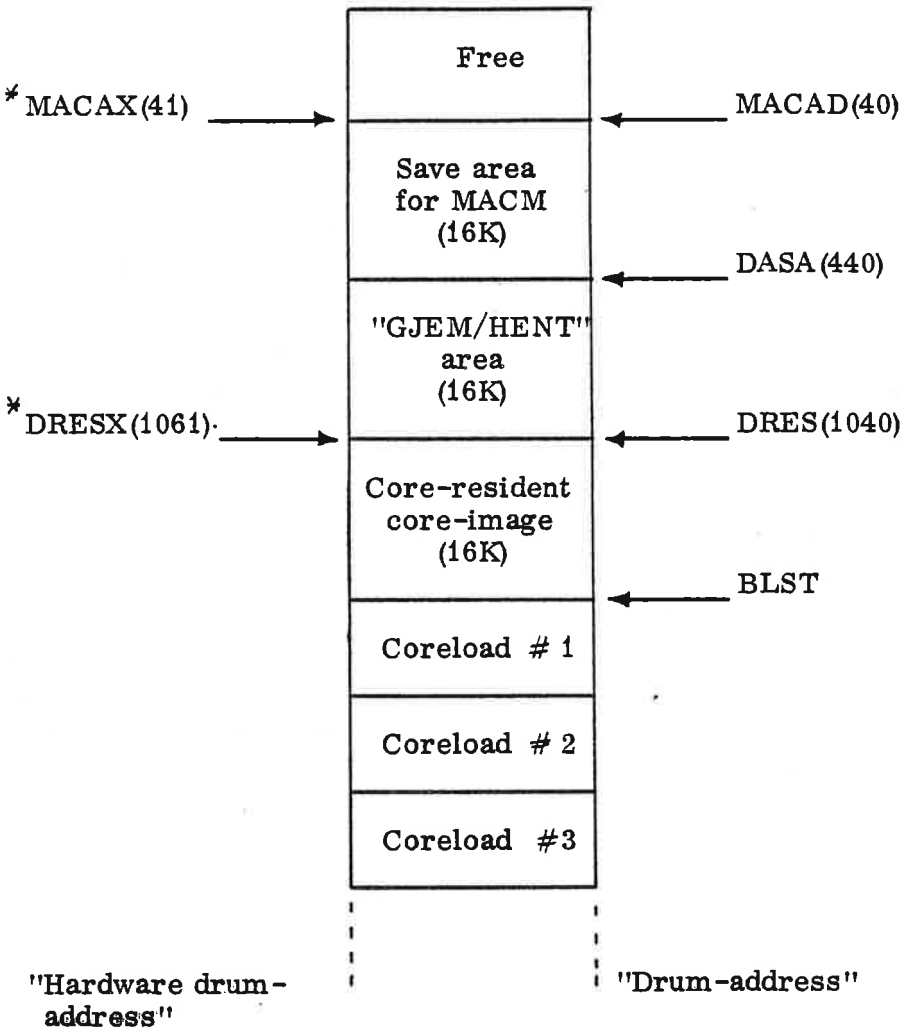


Figure 2. Example of drum-layout (The numbers in parentheses refer to a specific installation with 16K core)

For additional information see the manual "Generating SINTRAN II".

* MACAX = "Hardware drum address" corresponding to MACAD.

* DRESX = "Hardware drum address" corresponding to DRES.

8 GENERAL

The following software and documentation are necessary to run a MACM system.

a) A binary version of the MACM assembler called: DYNAMACM.

b) Symbolic User-Break-Point tape for drum called: UBP0.

c) Symbolic User-Break-Point tape for NCR-disc called: UBP1.

d) Symbolic User-Break-Point tape for CDC-disc called: UBP2.

The UBP tape must be assembled together with the user program. Before assembling the tape, 4 symbols must be defined:

```
DEVNO =
CRMAX =   } These symbols are described previously. The definitions
MACAD =   } must correspond to the last
DRES  =   } )9BYTT command.
```

e) Users unfamiliar with the MAC assembler need the manual: MAC Users' Guide (June 1971), in addition to this documentation.

9 SPECIAL PURPOSE COMMANDS FOR GENERATING SINTRAN III

Two new commands have been implemented:

```
)9SAVE␣A␣B␣C ↵        % CORE IMAGE TO DISC
)9GET ␣A␣B␣C ↵        % DISC TO CORE IMAGE
```

A number of 1K blocks is moved to/from core image and the segment area.

The three parameters must be defined MAC symbols, and the values are taken to be:

A	=	A core image address
B	=	Size (1K blocks)
C	=	Disc address (1K blocks relative to BLST)

Note that the commands)CLOAD and)CMOVE are not relevant when generating SINTRAN III.

NORSK DATA A.S.
Lørenveien 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

Publ. No. ND-60.009.02
November 1974

MACM
MAC Mass Storage Assembler

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM _____

_____

_____