COMASG Copy the MAS segments file COMASG

Purpose

MAS is an operating system, which is partly core resident and partly disc resident. While the core resident part can be created by using the Linkage Editor, some special actions must be performed to create or copy the segment file.
Two kinds of segment files exist:

- D:MASG is a DFM file of type UF residing in the directory of the System user. Although the file may have consecutive or non consecutive attributes, there may be no gap between the sectors.
- D:MSEG is a DAD with a sectorlength of 4082 bytes and a length that varies with the release that is used. It is created by the System Generation procedure %%DCICDC.

The D:MASG file is used for non CDC-SMD systems. When MAS needs a segment to read in it calculates the sector number in the D:MASG file where the segment starts and reads the segment into memory at address /300. For one segment, up to 10 sectors need to be read in.
The D:MSEG DAD is used for a system with CDC-SMD discs. These discs have variable sectorlengths and so it is possile to read a segment with one access to the disc.
To copy the D:MASG file from the starter pack to the user pack, the utility COMASG has been developed. This utility is the only way to fill the D:MSEG DAD and can also be used to copy a D:MASG file to another D:MASG file.

Calling sequence

The COMASG utility has to be called as follows:

```
ASG  FCOD=/40,FNAM=D:MASG[,USID=usid[,DAD=/Fx]]
RUN PROG=COMASG
```

Filecode /40 has to be assigned to the D:MASG file that has to be copied.
The DAD filecode /F2 must be assigned to the first DAD on the pack whereon the D:MASG file or the D:MSEG DAD to be filled resides.

Warning

COMASG does not give any error messages. The only way to know whether it performed successfully is the time it needs to run (2-3 minutes).

Purpose

To make a fast copy (5-10 minutes) from one CDC-SMD disc to another CDC-SMD disc or to certify such a disc. The COPY program is a stand alone program that can be loaded by answering the MONITOR? question of IPL-DK program with COPY.

Use of the COPY program

When loaded, the COPY program starts a dialogue:
- STANDARD INITIALZATION? (Y OR N)=

Replying '?' or any other character different from "Y" or "N" the standard values are output, being:

- CP (control panel) INT LEVEL = /07
- CU (control unit) ADDRESS = /16
- CU INT LEVEL = /20
- EMITTER DRIVE 0
- NOT A BIGD2 CU (so a BIGD)
- NO PRINT OF ALL STATUSES
- NO PRINT OUT OF THE MENU (the list of possible actions)

Replying "Y" will avoid to reply on the 7 initialization questions by using the prefixed ones mentioned above. The question 'WAITING FOR ACTION' (see below) is printed immediality.
Replying "N" permits to make ones own initialization by answering the following questions:
- CONTROL PANEL INTERRUPT LEVEL (1 HEXA CHAR)=

Reply the level of the control panel interrupt (normally 7) in one character. For all replies requiring a numeric input, the user should input hexadecimal characters without preceeding "/".
- CU ADDRESS (2 HEXA CHARS.)=

Reply the address of the control unit (e.g. 16 or 17). The control unit address in the lowest device address of the discs connected to it.
- INT LEVEL (2 HEXA CHARS.)=

Reply the interrupt level of the control unit (normally 10 or 20).
- EMITTER DRIVE NUMBER (0 OR 1)=

A drive number must be entered here. The drive number identifies the pack to be handled, for the COPY function, it is the pack that must be copied to another pack. The disc with the lowest device address is always connected to drive number 0 and the disc with the highest device address is always connected to drive number 1.
- IS IT A BIGD2 CU? (Y OR N)=

Reply "Y" (BIGD2 control unit) or "N" (BIGD control unit). The difference between the two control units is that a BIGD can address directly up to 128KW and a BIGD2 control unit up to 512KW.
- DO YOU WANT A PRINT OF ALL STATUS ERROR (FOR SERVICE PEOPLE ONLY) (Y OR N)=

Reply "Y" or "N", depending on your profession.
- DO YOU WANT A LIST OF POSSIBLE ACTIONS? (Y OR N)=

If "N" is replied, the message 'WAITING FOR ACTION' is output, if "Y", the following list appears:

```
POSSIBLE ACTIONS
0= COPY PACK TO PACK
1= BAD TRACKS LIST
2= DATA FAULT DETECTION
3= RESTART INITIALIZATION
4= NOT YET AVAILABLE
5= FLAG A BAD TRACK
6= PACK CERTIFICATION
7= PRINT VTOC
TO SELECT ACTION GIVE A CP. INT.
```

This list is followed by:
- WAITING FOR ACTION
which ends the initialization phase. On an erroneous reply of a question in the initialization, the question is re-output. On a valid (for the program) but incorrect (for the user) answer, the initialization can be restarted by action 3 or by a re-IPL.

Waiting for action, the program goes into an idle loop, waiting for a control panel interrupt. When this interrupt is given, the program outputs:
- ACTION=
and the user replies a number from 0 to 7, except 4 because that is not yet available, each starting an action. These actions are described below. Any action can be stopped by a CP interrupt, which causes the ACTION= to re-appear.

ACTION 0: COPY PACK TO PACK

This action permits to copy one disc to another one. Both discs must be CDC-SMD discs and connected to the same Control Unit. The copy takes 5-10 minutes (the Librarian SDD command 1-2 hours). The copy is only possible, when the emitter pack has a MAS structure. Packs to and from can be copied are:
- from 40MB to 40MB
- from 40MB to 80MB
- from 80MB to 80MB
- from 80MB to 40MB if the used space on the 80MB is not greater than 40MB.

To initiate the copy action, the following questions are output:
- RECEIVER DRIVE NUMBER (1 CHAR)=
Reply 0 or 1. It is the number of the unit to which the copy is to be made. If the Emitter drive is 0, the Receiver drive is 1 and vice versa. It is advised to copy from drive 0 to drive 1.
- RECEIVER DRIVE TYPE (40M=1 80M=2) =
Reply 1 or 2.
- IS YOUR PACK ALREADY CERTIFIED? (Y OR N)=
Reply "Y" implies that a certification (action 6) has been made in the past. In this case the copy process starts immediatily. When a bad track is detected on the Receiver pack, the program outputs its coordinates in the message:
- BAD TRACK DETECTED ON CYLINDER = xxx HEAD yy
This message is only an information to check, whether the bad tracks found now are the same as the ones found in a previous certification.
When the Receiver pack was not certified, so the previous question was answered with "N", the following message is output:
- DO YOU TAKE THE RISK? (Y OR N)=
Reply "Y" means that the user takes the risk to work with an uncertified pack. The copy process makes a quick premark before starting the actual copying. The reply "N" stops immediatily the action and the program goes into the 'WAITING FOR ACTION' state.

After these questions, the program prints the volume label of the Emitter pack, asks whether it is the right volume to copy and continues with asking for a volume name, a pack number and the date to construct the volume label of the Receiver pack. Then the copying starts with for each DAD the message:
- COPY OF DAD: xxxxxx
At the end of the copy process, the volume label of the Receiver pack is printed.

ACTION 1: PRINT OF BAD TRACK LIST

For this action, one additional question is output:
- EMITTER DRIVE TYPE (40M=1 80M=2)=
Reply 1 or 2, identifying the capacity of the disc.

Then the action starts. For each Bad Track, the message:
- BAD TRACK DETECTED ON CYLINDER = xxx HEAD NUMBER = yy
is printed. When all Bad Tracks have been printed, or when no Bad Tracks are present, the program turns into the idle state by printing:
- WAITING FOR ACTION

ACTION 2: DATA FAULT DETECTION

This action checks a disc on occurences of Data Faults. The tracks on which a Data fault is found are checked against the Bad Track list whether they are recorded. If not, an error message is printed and the disc must be re-certified. One additional question is output:
- DRIVE NUMBER (0 OR 1)=

Reply 0 or 1. This process performs the same actions as the copy process (action 0), without writing on an output pack. Per DAD the message:
- DATA FAULT DETECTION ON DAD xxxxxx
is printed.

ACTION 3: RESTART INITIALIZATION

This action allows to start the complete initialization of the COPY program, without re-loading.

ACTION 4

This action is not yet available. When called, an error message is output to inform the user about that.

ACTION 5: FLAG BAD TRACK BY OPERATOR

This action flags a track as bad and assigns an alternate track.
Two additional questions are asked:
- CYLINDER POSITION? (3 HEXA CHAR.)=
- HEAD NUMBER (2 HEXA CHAR.)=

With the replies to these two questions, the bad track is fully identified and flagged by the program. The track is read again to check the successfull completion of the operation. If no error is found, the messsage:
- BAD TRACK CORRECTLY FLAGGED
is output, if not the message:
- IMPOSSIBLE TO FLAG THE TRACK, SORRY, BUT THE PACK IS NO LONGER USABLE
is output, followed by the 'WAITING FOR ACTION' message.

ACTION 6: CERTIFY A PACK

For certification the dialogue is:
- IS YOUR PACK ALREADY CERTIFIED? (Y OR N)=

In case of a new pack, the reply must be "N". In case of an already certified pack, the reply must be "Y". When "Y", tracks flagged by a previous certification are kept and recalled as their recovery proceeds. The flagged bad tracks are printed as if they were found during this certification.
- FULL SURFACE MUST BE CHECKED? (Y OR NO)=

If "Y", all tracks of the disc are checked one by one (411 cylinders for 40M and 823 cylinders for 80M discs). If the reply is "N", the certification is restricted to the part of the disc specified in the answers on two questions, asking for the starting and ending cylinder to be certified (smallest part is one cylinder).

- LOOPING MODE? (Y OR N)=

This function permits to check the disc only once (reply "N") or more than once (reply "Y"). The experience proves, that two or three loops may be necessary. That is why it is advised to answer "Y" and to certify packs during night if possible. The certification process can be stopped by pressing the control panel interrupt button. A run indicator is printed at the end of each pass.

After the 'LOOPING MODE' question, the certification starts.

A loop consists of writing, re-reading and comparing each track:

- with pattern /0000 : 4 sectors of 2KW per track
- with pattern /FFFF : 32 sectors of 256 words per track
- with pattern /BFBF : 39 sectors of 205 words per track
- with random pattern : 64 secotrs of 105 words per track.

After certification, the disc is not yet usable. It should first be premarked (by a normal operator PK command). The premark asks whether the pack has been certified or not.

ACTION 7: PRINT VTOC

This action prints the VTOC (volume table of contents) of the disc.
Example:

DAD NAME	NB OF INT	NB OF SECT/GRAN	NB OF SECT/TRACK	SEC.LENGTH
SUPERV	0010	0008	0027	019A
D:CI	0003	0001	0004	1000

All values are output in hexadecimal format.

Purpose

Accessing a device, the hardware can return an error status, like parity error, data fault, etc. MAS retries such errors up to 5 times and, if it is a persistent error, /8000 is added to the status and it is stored in the calling ECB.
If the error is not persistent, no status is put in the calling ECB and the program continues as if nothing has happened. However, it might be very useful to know which device or (for disc) which track gives an error now and then. It might be the cause of a disaster in the future. Therefore, error logging has been developed. It logs the error, which was retried successully, on a file. The information in this file can be printed, and according to the obtained data actions can be taken to prevent real errors in the future.

Error logging in the system

Error logging is only active in the system, when the file D:ERLG, type UF is assigned in the system machine to filecode 33, thus:

```
ASG 33,DDFx,UF,D:ERLG
```

When during processing a hardware status is received, MAS records the error-information in a block in the System Dynamic Area, if filecode 33 is assigned. Every minute, the clock interrupt routine checks whether there are error logging blocks in the System Dynamic Area and, if so, it activates a MAS disc resident segment to write the block(s) onto the D:ERLG file. As this process is driven by the clock, the RTC must be switched on.
The D:ERLG is cyclic. When it is full, it starts from the beginning. A warning message is output, when the file is nearly full.

ERLOG utility

On the starter pack, a utility called ERLOG is delivered. With this utility one can create the D:ERLG file (which has a special format) and retrieve information from it.
The calling sequence of the utility is:

```
ERLOG
OPT FUNC=func[,DAD=dadfc][,USID=usid][,NBGR=n]
```

where:

func	is the function that should be performed: FUNC=CRE: create the D:ERLG file, FUNC=LIST: print the information recorded in the D:ERLG file.
dadfc	is the filecode of the DAD (/F0-/FF), where the D:ERLG file must be created (FUNC=CRE) or where it resides (FUNC=LIST). Default filecodeis /F0.
usid	is the userid where the D:ERLG file must be created or where it resides. Default userid is MASUP.
n	is the number of granules to be assigned to the D:ERLG file, when FUNC=CRE. Default is 1 granule.

Error messages:

```
INVALID OPTION STATEMENT (does not start with 'OPT ')
INVALID PARAMETER xxxx
NO FUNC PARAMETER
VALUE: xxxx OF PARAM: yyyy IS ILLEGAL
PARAM: xxxx IS REDUNDANT
ERROR DURING LKM: xx, STATUS= /yyyy
     (the utility uses several LKM's and LKM xx returned a status. For the
      meaning of the status, see Appendix C)
```

Output of ERLOG

For FUNC=CRE, the output of the utility (when it acted successfully) is:
ERLOG FILE CREATED
For FUNC=LIST the following items are printed:
- total number of recorded errors
- total number of recorded errors per device address
- information per error

The information per error consists of:
- device name and address
- machine name and name of the program that issued the error-causing access
- date and time when the error occurred
- hardware status
- for disc: cylinder, head and sector number
- number of retries
- control word given to the CIO start command.

Purpose

With FILEXC one can write/read load modules to/from a sequential file or device. FILEXC also handles files with other types (except EF), but these can also be handled by the Librarian.

Calling sequence

```
FILEXC
OPT FUNC=func,FCOD=fc,SYST=MAS,FNAM=fnam,FTYP=ft[,FDES=des][,KEEP=YES]
```

where:

func	is the function that should be performed: FUNC=IN reads the file from the sequential device to disc, FUNC=OUT writes the file onto the sequential device.
fc	is a filecode assigned to a sequential device.
fnam	is the name of a file.
ft	is the file type (UF, SC, OB or LM).
des	is a description to be put into the identification block. The maximum length is 16 characters, only consisting of alphanumeric characters and blanks.

- FILEXC assumes the file to be accessed in the current JOB Usid and DAD.
- KEEP=YES is only applicable when FUNC=IN. If specified, the file just read is kept in the directory of the current JOB Usid and DAD- SYST=MAS indicates that the utility is executed under the MAS operating system. The alternative is SYST=DOS.
- FILEXC only works in the Batch machine.

When FUNC=OUT, FILEXC first writes an identification block onto the sequential device (MT or TK). In this block, all parameters specified in the option statement are recorded. These parameters are checked against the option statement specified when the file is to be read in. After the identification block, the file is written.
FILEXC starts with printing a heading containing information about the file to be handled.

Error messages

```
OPTION STATEMENT MISSING
INVALID OPTION STATEMENT
INVALID KEY-WORD
TWICE THE SAME KEY-WORD
= NOT FOLLOWING THE KEY-WORD
KEY-WORD VALUE MISSING
SYST KEY-WORD MISSING
INVALID SYST KEY-WORD VALUE
FCOD KEY-WORD MISSING
INVALID FCOD KEY-WORD VALUE
FCOD FILE CODE NOT ASSIGNED
FUNC KEY-WORD MISSING
INVALID FUNC KEY-WORD VALUE
FDES NOT ALLOWED WHEN FUNC=IN
KEEP NOT ALLOWED WHEN FUNC=OUT
KEEP NOT ALLOWED WHEN SYST=DOS
INVALID KEEP KEY-WORD VALUE
FNAM KEY-WORD MISSING
```

```
INVALID FNAM KEY-WORD VALUE
FTYP KEY-WORD MISSING
INVALID FTYP KEY-WORD VALUE
KEEP NOT ALLOWED WHEN FTYP=OB
INVALID ASSIGNMENT TO A PERMANENT FILE
IMPOSSIBLE ASSIGNMENT TO A TEMPORARY FILE
MAS CONTROL IS NOT BATCH-MACHINE
I/O ERROR WHEN DELETING FILECODE
KEEP FILE OPERATION NOT SUCCESSFULL
EMPTY INPUT FILE
INVALID INPUT RECORD
INCONSISTENT INPUT FILE NAME
INCONSISTENT INPUT FILE TYPE
INCONSISTENT INPUT SECTOR LENGTH
INCONSISTENT INPUT GRANULE SIZE
TOO MANY INPUT SECTORS
INVALID INPUT SECTOR NUMBER SEQUENCE
TOO MANY/FEW INPUT RECORDS PER SECTOR
```

Purpose

IPL-DK is a utility that must reside in the 3rd granule of the disc from which is IPL-ed. It is a stand alone program which is loaded at IPL and gives the possibility to have more than one monitor on the same disc.

Behaviour

After IPL, the utility is loaded and started. It outputs the following questions:

- DAD:

Whether or not this question is output depends on the version of IPL-DK. The reply must be the name of a DAD residing on the IPL-ed disc, (default is the first DAD) or a question mark (?), which results in a print out of all the DAD names residing on the disc and re-outputting of the DAD question.

- MONITOR:

The reply on this question must be the name of a load module residing in the first Userid of the DAD indicated in the DAD question (e.g. SUP, LDMASR, COPY) When MAS8 (so LDMASR) is intended to be loaded, the DAD question must be replied with the first DAD of the disc, the same applies for a DOS monitor. A question mark (?) results in a print out of all load modules residing in the first Userid of the DAD and re-outputting of the MONITOR question. A <CR> letsthe DAD question re-appear.

- LOAD ADDRESS:

Reply the address where the monitor is to be loaded. Default (and normal input) is a carriage return <CR>, being address /0000.

Purpose

LDMASR is a utility to load a MAS8 (extended mode) monitor, because such a monitor cannot be loaded by the normal IPL.

Behaviour

After loading LDMASR asks for the name of the MAS8 monitor to be loaded:

- MONITOR:

The reply must be the name of a load module containing a MAS8 monitor, which resides in the first Userid of the first DAD of the disc from which is IPL-ed. The default (<CR>) is MASR.

Purpose

With OVLGEN, one can move a load module, from filecode /D6 (/L) to the D:MASG file. The D:MASG file must be assigned to filecode /40.
A full description of this utility is given in Appendix C with LKM 47.
The OVLGEN utility only runs in the Batch machine.

0.25M Flexible (Floppy) Disc Driver

0.25M Flexible discs connected to a 0.25M flexible disc Control Unit are accessed as a physical device with direct access at sector level. Every sector is available for data storage.

The logical and physical characteristies of the flexible discs are as follows:

- Physical sector length is 128 characters.
- Logical record length using 1 LKM instruction (4 sectors):
  a) connected to a programmed channel: up to 512 characters
  b) connected to the I/O Processor: the user may read or write to a maximum of 1 track (3328 chars).

  If a record comprises more than one physical sector, it is defined by the physical address of the first sector in the record.
- A track is 26 sectors, or 3328 characters (not interlaced).
- A floppy disc has 77 tracks (2002 sectors, 256256 characters).
- A sector is defined by a sector number from 0-2001 (the sector number given in word 5 of the ECB used for I/O operations with LKM 1).
- Bad Track conditions are handled by the driver.
- The Floppy Disc Driver package is included during system generation, in the same way as other physical devices, using the mnemonic `FL'.

Floppy Disc Operations

The order loaded into A7 before issuing an LKM 1 determines which of the various possible functions is carried out by the control unit. These can be summarised thus:

Order	Operation
/00	Get extended information about a filecode
/11	Read Sector
/15	Write Sector
/3A	Compound Read
/3B	Compound Write
/3F	Write Sector and Verify
/2F	Write Deleted Data Address Mark
/3D	Write Deleted Data Address Mark and Verify
/3E	Search Key
/3C	Search Key Masked
/2D	Door Lock
/2E	Door Unlock
/30	Return Information about a Filecode.

The ECB Layout

The ECB address is loaded into A8 before issuing the LKM 1 requests. The layout of the ECB is as follows:

	0 7	8 15
Word 0	Event Byte	Filecode
1	Buffer Address	
2	Requested Length	
3	Effective Length	
4	Returned Status	
5	Sector Number	

where:

- 'Filecode' is the filecode assigned to the floppy disc unit.
- The 'event byte'; bit 0 is set to 1 by MAS, to indicate completion of the operation.
- Buffer Address is the user's buffer address, which can contain or receive data as normal, but can also contain control information (e.g. Search Key or Write Deleted Data Address Mark).
- Requested length is expressed in characters, and should be even (except for search key operations); MAS will round down odd-numbered values.
- Return Status codes are summarised later.
- Effective length is the actual message length, in characters, read or written.
- Sector Number is a relative sector number or sector address, in the range 0-2001.

Read/Write Operations

/11 - Read Sector

From 1 - 4 sectors (each of 128 words) may be read in one operation; thus the maximum record length is 512 characters. The sector number of the first sector in the record determines the start address or sector number specified in word 5 of the ECB.

/15 - Write Sector

From 1 - 4 sectors can be written directly. The requested length should be even and not greater than 512 characters.

/3F - Write Sector and Verify

This order is performed as for /15, but with the additional feature that the disc revolution following the write operation is used to read and verify the record.

/3A - Compound Read

This order allows the user to read up to 3328 characters (i.e. 1 track or 26 sectors) in one disc revolution. The starting sector address quoted in ECB word 5 is the logical start of the track.

The operation is performed correctly with the disc control unit attached to the Input/Output Processor (IOP). If the disc control unit is attached to the Programmed Channel the command is accepted, but more than one disc revolution is required to complete it. The ECB values are as for function /11 (Read), except that the allowed requested length is in the range 2 - 3328.

/3B - Compound Write

This order allows a complete track (26 sectors) to be written in one revolution of the disc, as long as the disc control unit is connected to the IOP. If it is connected to the Programmed Channel, it will require more than one revolution. The requested length is in the range 1 to 3328 characters.

/2F - Write Deleted Data Address Mark (DDAM)

This order allows the user to write a special pattern on the disc. This pattern should be placed in the user's buffer before the LKM 1 is issued. The sector number to which the mark is to be written is entered in word 5 of the ECB, while word 2 of the ECB contains the requested length expressed in characters.

If more than one sector is to contain the DDAM, the requested length is in the range 128 - 512.

/3D - Write Deleted Data Address Mark and Verify

This is carried out as for order /2F, but a further revolution of the disc is used to read back the pattern and verify that it has been written correctly.

Search Key Orders

/3E - Search Key

This order allows the user to search for a pattern of characters which begin at the start of a sector. The search is carried out within and including a start sector number and end sector number, specified by the user.

Either an even or an odd number of characters may be specified as the search pattern. This pattern is placed in the user's buffer, and is immediately followed by the start and end sector number.

The length of the pattern is specified in word 2 of the ECB.

After a successful search, the sector number of the sector containing the pattern is entered by MAS in word 5 of the ECB.

The following diagrams illustrate how the user sets out his buffer before issuing the LKM 1:

a) For an even number of characters:

1	2	length of search pattern
3	4	character string = 6
5	6	
Start sector number		search limits
End sector number		

b) For an odd number of characters:

1	2	length of search pattern
3	4	character string = 5
5	ignored	
Start sector number		search limits
End sector number		

/3C - Masked Key Search

This is conducted in the same way as the Search Pattern order /3E, except that the user is able to blank or mask certain characters within the pattern string which will be ignored during the search operation. The character positions to be masked are filled with the space character (/20).

The following example shows how the user should lay out his buffer before issuing the LKM 1 request.

Suppose the pattern to search for starts at character 3 of the sector, is 7 characters long and consists of the characters A and B at character positions 3 and 4 and the character C at position 6:

```
/20             /20
/20             A               search pattern
B               /20
C               ignored
Start sector number             search limits
End sector number
```

Other Orders

/2E - Door Lock

This command locks the loading door of the floppy disc unit, preventing manual opening of the door until the Door Unlock command (/2D) is received. The ECB should be initialised as follows:

```
0                       Filecode
1   Buffer Address (dummy value)
2   Requested length (dummy value)
3   Returned Length (unchanged)
4   Returned Status (updated by the system)
5   Sector number (dummy value).
```

/2D - Door Unlock

This order unlocks the door of the floppy disc drive unit, allowing the operator to open the manual door latch in order to extract the disc. The ECB layout is exactly the same as that of the /2E (Door Lock) command.

/30 - Return Information About a Filecode

This command causes the unit to return a value for Best Length to word 3 of the ECB. The other locations are as described in Appendix C under LKM 1.

Error Messages

If an I/O operation is attempted on a drive which is inoperable (e.g. the door is open), the following message is printed on the console:

```
PU FLxx,yyyy,RY
```

where:

xx is the Floppy Disk Address
yyyy is the status
RY means Retry or Release Device.

For the retry the operator must remove the cause of the error, when the I/O operation should proceed automatically.

Returned Status for Flexible Disc

If bit 0 is 0 but bit 1 is 1 a hardware error has occurred, and the following bits have significance:

Bit	Meaning
2	Key not found
4	Deleted data mark read
5	Record not found
6	Write protected
10	The request was completed after one or more retries.
11	Program error
12	Incorrect length
13	Data error.

If bit 0 = 1 and bit 1 = 1, an error occurred in the calling sequence. If bit 11 is also set, the request code is invalid or the sector number is incorrect.

1M Flexible Disc Driver

1M (1 megabyte) flexible discs are supported by the FL1MZ and the FL1MB control units. These control units are always connected to the IOP and they can handle 1M flexible discs as well as 0.25M flexible discs.

Disc types

This flexible disc driver supports the following flexible disc types:

- F1 : 0.25M flexible system disc
- F2 : 0.25M flexible data disc (MAS relase 8: type F6)
- F3 : 1M flexible system disc
- F4 : 1M flexible data disc
- F5 : 1M flexible data disc

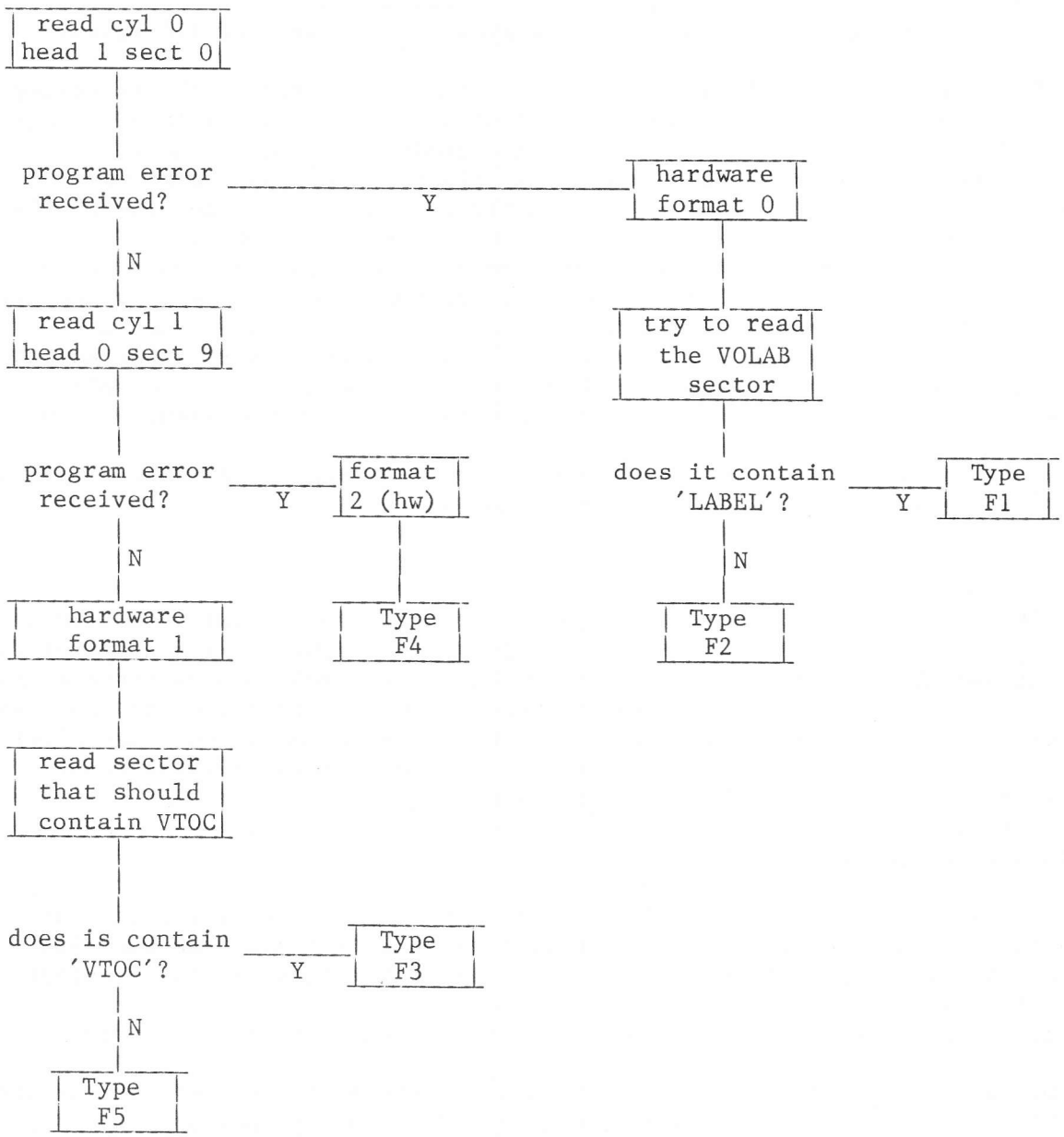
Hardware formats

There are three hardware formats for flexible discs.
The characteristics for each format are:

Hardware characteristics	Format 0	Format 1	Format 2
number of surfaces	1	2	2
number of cylinders	77	77	77
sect/track cyl 0, head 0	26	26	26
sect/track cyl 0, head 1	-	26	26
sect/track other cyls	26	26	8
sectorlength (char) cyl 0, head 0	128	128	128
secl cyl 0, head 1	-	256	256
secl other cyls	128	256	1024
number of useful cyls	74	74	74

The translation of the hardware format into the flexible disc types F1, F2, F3, F4 or F5 is done at IPL time, if there is a flexible disc in the drive, or when a ready interrupt is received.

The assignment of the types is done according to the following scheme:



A program error is returned by the control unit, if an access is made to a non-existing track or to a non-existing sector (i.e a too high sector number). So the first check on program error determines whether there are one or two tracks (surfaces) available on a cylinder and the second check whether there are 8 or more (so 26) sectors available on the track. These two checks identify fully the hardware format.

Orders

For this driver, the following orders are allowed:

- /01 or /11: read; allowed for all types of flexible discs.
- /05 or /15: write; allowed for all types.
- /13 : verify; allowed for all types.
- /21 : read VTOC; allowed for type F3, simulated for type F1.
- /25 : write VTOC; allowed for type F3, simulated for type F1.
- /2D : door lock; allowed for all types.
- /2E : door lock; allowed for all types.

- /2F : write deleted data; allowed for types F2, F4, and F5.
- /3A : compound read; allowed for all types.
- /3B : compound write; allowed for all types.
- /00 or /30: get filecode/device information; allowed for all types.

- For read/write on DAD or disc level, with type F1 and F3 the requested length must be the length of one logical sector. For Data flexible disc, any length is accepted and there is an automatic chaining on next tracks.
- For read/write VTOC, the ECB must be filled as follows: ECBSC must contain 0 and the word ECBHD must contain the physical sector address of the VTOC (this address can be found on displacement /52 of the VOLAB).
- With verify, the command of the buffer in memory is compared with the contents of one or more sectors on the disc. It can be used after a write, to check the written data. The buffer must remain the same between the two commands. If the comparison is incorrect, the status Data Fault is returned.
- Door orders are only executed when the addressed drive is operable and when the 'door lock' option is present. In other cases, the command has no effect.
- Write deleted Data, writes the sectors with deleted data address marks. The ECB must be filled as in the write command.

Bad track handling

The FL1MZ and FL1MB controllers use cylinders 75 and 76 to assign bad tracks. These cylinders cannot be accessed by an application. This is a constraint in the transportability from MAS to other systems (like IBM). Others systems may use cylinders 75 and 76 to put data on, which cannot be retrieved by MAS. This data can even be overwritten, when a bad track is detected by the controller.

The driver itself does not contain bad track handling, so if an error occurs, the flexible disc should be exchanged.

Flexible disc type F1

The type F1 is a 0.25M flexible disc, with an X1215 disc simulated lay-out. No DAD's can be declared on it. The whole disc is seen as containing one DAD, called DADFF1. No VTOC resides on the disc, it is simulated by the flexible disc driver.
The disc type, stored in the VOLAB of the type F1 flexible disc is 'FLD1'.

One logical sector, being 410 bytes, is stored physically in 4 sectors on the flexible disc. An interlace table is maintained in memory (not on disc, because it has no VTOC). The interlace factor is 2.
Interlacing is not performed on logical setors 0, 1 and 2. For logical sector 0, the sectors starting at physical sector 0 are accessed. For logical sector 1 sectors from physical sector 4 are accessed and for logical sector 2, accessing starts at physical sector 12.

System commands like ASG and DUF, on a type F1 flexible disc are possible. However some commands concerning DAD's like DCD and DLD (declare and delete DAD) are not possible, because of the DADFF1 VTOC simulation.

The type F1 is considered by MAS as a disc and so a filecode should be assigned to it during System generation. Thus:

FCD: /Cx,MFyy

/Cx is the filecode assigned to the type F1 disc in the System machine.
MFyy is the device mnemonic for the disc (MF) and its device address.
Note that MF is only the device mnemonic during System generation. In the monitor the device is considered to be a disc, with the device mnemonic 'DK'.

A declaration of the disc in a machine declaration is done as follows:

```
FCD /Cx
FCD /Fy,/Cx,DADFF1
```

/Cx being the filecode assigned during System generation and /Fy the filecode to be assigned to the (only) DAD on the disc.

Receiving a ready interrupt of a type F1 disc the message:

```
FLOPPY DATA F1
```

is output by the system.

A type F1 disc must be premarked. The premark asks for the type of flexible disc: FLOPPY TYPE F1 OR F3

For this type, the reply must be F1. Premark does not ask question about the DAD to be created.

Flexible disc type F2

A type F2 flexible disc is a Data disc with hardware format 0. It can only be accessed on physical level. So no ASG a logical file, DUF and other system commands are allowed for this type of disc. Although the disc is recorded as 'DK' in the system, one may assign any filecode to the disc, e.g. ASG /10,DKxx.

Receiving a ready interupt, of a type F2 disc the message:

```
FLOPPY DATA F2
```

is output by the system.

Flexible disc type F3

The type F3 flexible disc contains a DAD structure. The first DAD starts at cylinder 1 and the lay-out of the disc is as follows:

- Cylinder 0: Physical sectors 0, 1, 2 and 3 are IPL sectors.
- Cylinder 1: Logical sector 0 contains the volume label.
  : Logical sectors 2-5 contain the catalog of the first DAD.
  : Logical sector 6 is the not used Bad track sector.
  : Logical sector 7 contain the VTOC.

The type F3 is a hardware format 1 disc with a X1215 simulated structure. DAD's can be declared, files assigned etc. DAD's have interlace factors, which must not have a common divisor with the number of sectors per track (i.e. 26). Interlace factor 1 is allowed.

The disc type stored in the VOLAB for the type F3 disc is 'FLD2'.

Logical sectors contain 410 bytes. The first cylinder of the disc cannot be accessed. For each access to the disc, the calculated cylinder number is incremented with 1, so logical sector 0 is translated into an access to the first sector on cylinder 1.

Generating the flexible disc, in System generation a /Cx file code must be assigned to the device (see type F1)

Receiving a ready interrupt, the system outputs:

```
FLOPPY DATA F3
```

when this type of disc was mounted.

A type F3 disc must be premarked and the question:

```
FLOPPY TYPE F1 OR F3
```

must be answered with 'F3' for this type.

Flexible disc type F4

The type F4 flexible disc is a Data disc with hardware format 2. For this type, the same rules apply as for the type F2 flexible disc. Note that the sectorlengths for cylinder 0 head 0, cylinder 0 head 1 and the other cylinders differ.

Receiving a ready interrupt, the system outputs:

```
    FLOPPY DATA F4
```

when this type of disc was mounted.

Flexible disc type F5

This is a flexible disc with hardware format 1. The same rules apply as for the other data disc types. And of course the message:

```
    FLOPPY DATA F5
```

is printed when this type of disc gave a ready interrupt.

GENERAL

This Appendix describes how the user can incorporate his own user driver into the MAS Monitor. No alterations to existing Monitor routines are required, andthe driver package is then included during the system generation procedure.

PREPARATION OF THE DRIVER PACKAGE

After being coded, the driver is compiled and the object module is stored within the userid MASGEN in the object library MASOB for non Extended Mode systems and in the object library IOCSOB for Extended Mode system. This can be achieved by using the Librarian processor.
A driver consists of two parts:

- the device dependent LKM 1 (I/O) handling, filling the device dependent part of the DWT (device work table) and executing the WER instructions - for devices connected to IOP - and the CIO start instruction.
- the interrupt routine, which handles the interrupts, executes the SST, INR, OTR and CIO stop commands and branches upon end of I/O to the general end of I/O handler of the system.

Normally, both parts are coded in the same module. This module contains two entry points, one for the device dependent I/O handler (called D:<name>) and one for the interrupt routine (called S:<name>).

The driver must be made known to the system by including its DWT in the DWT chain and by filling the address of its interrupt routine in the appropriate interrupt location in the LOCAT module.

The LOCAT module

The incorporation of the driver in LOCAT (so filling the interrupt location) can be achieved by answering the "USER INT:" question in the System generation CONGEN process with <lev>,<name>. The System generation will then at interrupt location <lev> store the address of the external I:<name>. An EXTRN for I:<name> is also generated.

The DWT chain

For each device, a DWT must be incorporated in the DWT chain. For disc devices, also a DCT (dics control table) must be included in the DCT chain, but as the writing of disc drivers needs special knowledge of the MAS system, it is not discussed here.
A DWT can be put in the chain by replacing the module USDWT, which is present in the MASOB resp. IOCSOB library by an own written USDWT module.
The standard USDWT module consists of one instruction:

```
USDWT    EQU  0
```

As the last chain address in the DWT chain points to USDWT it contains zero (i.e. end of chain) when the standard module is used. Including a user written DWT in USDWT, the last chain address value changes from zero to the address of the DWT to be added, and that is exactly the intention.

MASLIB

EXAMPLE

The lay-out of USDWT, containing one user written DWT is:

```
          IDENT     USDWT
          ENTRY     USDWT
          ENTRY     I:<name>        address stored in the interrupt location in
                                    LOCAT.
          EXTRN     D:<name>        device dependent I/O handler.
          EXTRN     S:<name>        interrupt handler.
I:<name>  MSR       8,A15           save 8 registers in the system stack.
          LDKL      A6,USDWT+2      Fill A6 with the DWT address. Note that the
                                    DWT starts at the <dev name> word!
          ABL       S:<name>        branch to the interrupt handler
USDWT     DATA      NXTDWT          address of the next DWT, if no DWT, 0.
          DATA      '<dev name>'    mnemonic of the device that is included.
          DATA      <dwtda>         flags and device address.
          DATA      <best length>
          DATA      D:<name>        driver address
          ---                       ) For a description of the DWT, the user
          ---                       ) should refer to Volume IV, the
          ---                       ) Trouble Shooting Guide.
          DATA      ---             last word of the DWT
NXTDWT    EQU       0               or the start of the next DWT.
          END
```

Contents of the DWT

The DWT is described in the Trouble Shooting Guide. Here is indicated:
- the fields that must be filled during DWT definition
- the fields that are filled upon entering the driver
- the fields that must be filled by the driver, returning to the end of I/O handler

Defining the driver

The following fields must be filled, defining the DWT:

-DWTDN	device name
-DWTDA	device address and flags
-DWTBLG	best length
-DWTDRV	driver address
-DWTRY	last 6 bits: the device type
-DWTC:N	address of a word containing /8000 (not busy)
-DWTSST	address of I:<name>+2
-DWTFLG	bit 4: transfer per word (1) or transfer per character (0) bit 5: single device controller (1) or multiple device controller (0) bit 6: device connected to IOP (1) or programmed channel (0)

By the general LKM 1 handler (X:IO) are filled:

-DWTBUF	buffer address in System Dynamic Area
-DWTRLG	requested length to be transferred
-DWTORD	the last 6 bits of the order
-DWTA5	PCT address of the calling program
-DWTA6	scheduled label address in the calling program (0 if no Schlab)
-DWTATT	used internally in X:IO
-DWTDET	used internally in X:IO
-DWTUEC	user's ECB address
-DWTURO	user's order
-DWTNT	used internally in X:IO
-DWTIME	used internally in X:IO

-DWTQUE used internally in X:IO
-DWTFCT address of the FCT entry in the System Dynamic Area

Fields, not used by the general MAS I/O routines can be used freely by the driver. The driver should, when input is received from a programmed channel device, store the current number of received characters in the field DWTEFL. The field DWTBUF must then always contain the address in the buffer where the next character is to be stored. The minimum length of the DWT must be /46 characters (including the chain word). There is no maximum.

Inputs and outputs

On entering the driver from the general I/O handler (X:IO), the DWT is filled as indicated above. Furthermore:

- A6 contains the DWT address i.e. points to DWTDN.
- A4 contains the last 6 bits of the user order
- the MMU of the calling program is loaded.

On end of I/O, when the driver should return to the general end of I/O handler, the following registers must be filled:

- A6 with the DWT address
- A8 with the address of DWTIOB
- A2 with the status received from the SST instruction
- A3 with the effective length.

After filling these registers, a branch must be made to the external R:TUR1:

```
ABL     R:TUR1
```

MAS routines

Some MAS routines can (or even must) be called from the driver.

- M:DIS1: the dispatcher. This routine has to be called from the device dependent I/O part of the driver to return. M:DIS1 expects 8 registes in the A15 stack, which are already stored there upon entering the driver. The routine has to be called with an absolute branch:

```
ABL     M:DIS1
```

- M:LRTN: absolute return routine. When the interrupt part of the driver decides not to return to the end of I/O handler (because the I/O has not ended)this routine must be invoked. M:LRTN expects 8 registers in the A15 stack, which are stored there by the I:<name> routine. M:LRTN is invoked by an absolute branch:

```
ABL     M:LRTN
```

- R:GO62: go to hardware level 62 routine. When an interrupt is received from any device, it enters the interrupt handler on a hardware level equal to the interrupt level of the device. Only interrupts with a lower interrupt level are serviced, as long as the interrupt routine runs at the level entered. Therefore the routine should go as soon as possible to hardware level 62 to allow other interrupts to come. R:GO62 is called, using A15:

```
CF      A15,R:GO62
```

Note that, entering an interrupt routine, the system is in inhibit mode. The driver must issue an ENB (enable) instruction as soon as possible to let the system re-get the normal execution.

- Execute: hardware instruction routines. To perform hardware instructions, the driver may use some general routines. The routines must be called with an absolute branch, the return address must be loaded in A4. The instruction is constructed in A3 and the driver can supply a control word in A2. A1 is destroyed. Calling sequence:

```
      LDKL    A4,ret
      ABL     routine
ret   ----
```

The routine is S:TIO for a start I/O instruction, S:SST for an SST, H:LTIO for a halt I/O, T:TST for a TST, I:NRIO for an INR and O:TRIO for an OTR instruction. On return the condition register is the condition returned by the hardware instruction.

- C:INPT: check input character. This routine executes the INR instruction for a programmed channel device (input), it checks the received character on special functions (backspace, delete line, end character) and performs a CIO halt instruction if necessary. On input A6 must contain the DWT address, the routine does not return to the driver. Calling sequence:

```
ABL     C:INPT
```

A MAS manual is not complete without some words about the internal structure of the monitor.
In the monitor there are a lot of LKM routines present. They are invoked by the LKM interrupt handler (I:LKM) via the table T:LKM. This table is generated during Sysgen and may be examined by the one (or two) who wants a somewhat deeper knowledge about this subject. Other information can be retrieved from the description of the LKM 47 (user written LKM) in Appendix C.

Furthermore, the monitor is more or less a Foreground machine called SYSTEM. In a Foreground machine, programs are running and indeed, that is also the case for the System machine. All programs have a PCT (program control table) and are chained decently just like the PCT's in a normal Foreground machine. In the System machine also per Foreground machine a PCT is present to run the FCL task of that Foreground machine.

Programs running in the System machine

A list of the so called system programs may be obtained by giving a MAP command in the System machine. The system programs run on the lowest software level (= highest priority) in the system, except X:IDLE which runs on the highest available software level.
The system programs are:

----- X:IO LKM 1 (I/O) handler -----

X:IO performs all device independent functions of the LKM 1. For the start of an I/O it is entered by I:LKM, for the end of I/O it is entered by the driver.
Functions performed by X:IO are at start of I/O:
- check the validity of the ECB
- fill the DWT (see Appendix F)
- ask an intermediary buffer in the System Dynamic Area for non-disc devices and (if output) move the user buffer to that area.
- set the device to busy. If the device is already busy, create a device request queue.
- open a spool file or transform a request to a spooled device into a request to a file on the spool DAD.
- call the "device dependent" routine:
    - for a request to a physical device: the driver
    - for a request to a logical disc file: disc file management (M:DFM)
    - for a request to TDFM: the TDFM package
    - transform a request to a DAD into a request to a physical disc device.

Functions performed at end of I/O are:
- fill the user's ECB with the returned status and the effective length
- set the device free or take the next entry from the device queue and start it
- move the buffer to the user area (if input) and free the intermediary buffer, if that was asked during start of I/O
- clear the DWT
- set the event on the user's ECB
- start the scheduled label (if any).

----- X:MASG MAS segment handler -----

The disc resident segments of MAS are read in and executed under the X:MASG program. At system initialization, is determined whether a D:MSEG DAD (system filecode /FF) or a D:MASG file is to be read and according to this information, an ECB is constructed, used by X:MASG to read the segments. Note that, if D:MASG has to be read, the read is done at DAD level (/F0). The start sector

of D:MASG and the calculated start address of the segment to be read are added and from that sector, the segment is read. This means that the D:MASG file must consist of consecutive granules, although it needs not to have the "CONS"attribute.
Upon activation of X:MASG, A3 contains in the first 6 bits an entry number in the segment. The segmentnumber is stored in the last 10 bits of that register.

Functions performed by X:MASG are:
- check whether the segment to be called is already in core. If not, read it from disc. The current loaded segment number is stored in the field T:SCUR, which is an entry that can be found in the MAS Link list.
- check the result of the I/O operation to read the segment, if error is returned, abort the system with error /000B, except when the error was "disc not operable". If not operable output the message:
  SYSTEM DISC NOT OPERABLE, RESTART IT
  on system filecode /EF and stop the system until the system disc gives a ready interrupt.
- start the segment by an indirect branch to the entry (present in Link list) T:OVLA+(2*entrynr).

----- X:SWIO swap handler -----
Swapping in MAS is done via this task.
Functions performed are:
- determine whether the swap DAD (system filecode /F1) is a DAD with small (i.e 410 bytes) sectors or with large (4092 bytes) sectors. Construct an ECB according to this information.
- For swap out, write the program to the D:CI DAD, to a file, starting at the sectoraddress recorded in the program's PCT in word PCTSWN. At least, when the program has been declared swappable, for read only programs, writing is not performed. Then free the pages, occupied by the swapped out program.
- For swap in, reserve pages in the Dynamic Loading Area. The number of pages to be reserved is recorded in the program's PCT, word PCTREG. Then read the program into the reserved pages from the D:CI DAD. The address where the program resides is for read-only- or not previously swapped out programs PCTSW1 and for previously swapped out programs PCTSWN. Then start the actions recorded in the program's PCTMOV queue, which are delayed, because the program was loaded.

----- X:ALGR allocate/free granules -----
The (de)allocation of geanules has to be maintained by a separate task, to prevent that two tasks request granules at the same moment. This could lead to allocation of the same granule to more than one file. With a separate task, each system program that requests granules, activates X:ALGR and, if busy, an activation queue is created so that only one request is handled at the time.
On entrance of X:ALGR, A3 contains an entry number:
- 0: allocatate granules
- 2: free granules.

A4 contains the address of a control block. Lay-out:

bit	0	1	15
word 0	E		NC
1	A4		
2	DAD address		
3	# of granules		
4	status		
5-n	granule address		

where:

E — is an event bit, set upon completion of X:ALGR

NC — is an indication for consecutive (NC=0) or non-consecutive (NC=1) granules.

A4 — is the address of the control block

DAD — is the address of the DAD control table in the System Dynamic Area. In the DAD control table, a BITTAB is recorded, wherein X:ALGR puts its alterations.

gran — is the number of granules to be allocated or freed.

status — is zero if the requested number of granules could be allocated and 1 if a DAD overflow (not enough free granules) occurrud.

granad — is for consecutive granules a one word area containing the start address of the granules to be allocated/freed. For non-consecutive files, it is an n-word area (n is the number of granules to be allocated/freed). Each word containing one granule address.

----- X:TDFM segment handler for TDFM -----

X:TDFM performs about the same actions as X:MASG for TDFM segments under Extended Mode systems.
For TDFM, some segment buffers are reserved over 32Kw. X:TDFM checks whether the requested segment has already been loaded in one of these buffers and, if so, starts it. If not yet loaded, the least recently used segment is overwritten by the required segment and started.
X:TDFM reads it segments from the monitor load module (MASR) and not from the D:MSEG/D:MASG segment files.

----- X:USVC user service calls -----

X:USVC is used to start core resident "segments". It is activated with anentry number in A3. This entry number is searched in the table T:RMAC (which is generated during Sysgen) and it branches to the associated label. This task is used to activated core resident LKM's as specified in Sysgen.

----- X:LPxx spool out task -----

This task performs unspooling of files to the lineprinter (xx is the lineprinter's device address). It is activated by a segment that closes the spool file via a write EOF.

----- X:CRxx spool in task -----

This system task reads a file from the spooled cardreader (device address is xx) and writes this file to a file on the spool DAD (D:SPCR).

----- X:DUMP dump memory -----

This task is activated when a dump is required of (a part of) the memory. The dump is made via a system task, because dump commands are handled in disc resident segments (they are not often used). When the dump was made in a segment, under X:MASG no other segments could be activated and because dumping is a time consuming activity, it would block the system.

----- X:RTC real time clock handler -----

This task is only activated, when programs are connected to a timer or clock. It is done in a separate task, because this action is time-bounded and to free the clock interrupt routine.

----- X:OCOM operator command handler -----

X:OCOM is activated by the control panel interrupt routine (I:CTPN), when the INT button is pushed. It outputs an "M:" on system filecode /EF and reads then the operator command and activates the program that handles the given command. Furthermore it is used by system task to print a message to the operator, so that the calling task is not blocked during the time the message is being printed.

----- X:IDLE idle task -----

This task with the highest software level so the lowest priority is always eligible to run, but because of its low priority. It is only running when no other program wants to run, because it is not active, waiting or suspended (on resources, LKM, etc)
This task consists only of two instructions:

```
X:IDLE    DLC  31        dummy instruction
          RB   X:IDLE
```

MAS abortions

MAS sometimes aborts, due to hardware, software, or even user errors. It is impossible to specify here, what the user should do when a system abort occurs. However some hints can be given.

The stand alone dump

When MAS aborts, a stand alone dump (if generated) is output on the lineprinter. The dump consists of a print out of the current registers, the current MMU register and the whole memory. In such a dump, some area'a can be indicated that may be of help for debugging.

-- P:CUR --

P:CUR is a one word area in the LOCAT module. It is an entry that can be found in the MAS link list. It contains the address of the PCT of the program that is currently being executed. At system abort it might contain the (system) program that caused the abortion. However, as the current program can be interrupted from a driver interrupt routine (which does not change P:CUR) it is not always true.

-- T:SCUR --

This is the current (or last) segment, executed by X:MASG. It is not in the scope of this manual to give a description of all segments (currently 107 arepresent), but a list of the segments with their functions performed can be obtained from the SSS department.

-- The A15 stack --

The A15 stack is (when generated normally) an area in the LOCAT module, ranging from /100 to /300. It is filled from the high address (/2FE) to the low address (/100). A stack overflow results in a system error /0007. The address of the first free word in the stack (i.e. the free word with the highest address can be found in A15).

-- The System Dynamic Area --

In the System Dynamic Area all buffers, control blocks etc used by the system are put. The start address of the SDA can be found in the system's link list, entry SYSDYN. All blocks in the SDA are chained via the first word. When the chain word is odd, the block following it is free, if even, it is occupied.

Error codes

Only abort codes, that occur relatively frequent are discussed here.

--- System error /0000 ---

This error code means: unknown interrupt. A hardware interrupt was received on an interrupt level that was not generated into the system. This level may be found by taking the word addressed by the contents of A15 plus two. This word contains in the first 6 bits the unknown interrupt level.
The same error can occur, when the system branched erroneously to an address in the interrupt locations. In that case, the interrupt level will normally be 61, 62 or 63. No standard debugging for this cause of the error can be given.

--- System error /0009 ---

This error code means: chain in the System Dynamic Area is corrupted. It is only detected, freeing a block from the SDA, when the block to be freed is not in the chain, or when it is already free.
The address of the block that was to be freed can be found as follows: take the current stack address (A15) and add 18. At this address, the address is put, that was to be freed, but that could not be found in the SDA chain.

--- System error /000B ---

The system aborts with error code /000B, when X:MASG has read a segment, but the event returned a status. The status (which is returned by an LKM 1) can be found via the register A8, which contains the address of the X:MASG ECB.

--- System error /0015 ---

This error means: invalid instruction in the monitor. The address of the word, containing the invalid instruction can be found via register A15. The contents of this register plus 24 gives the address of a word, containing the address of the invalid instruction.

--- System error /001B ---

Error /001B means that there was an invalid attempt to free granules in a DAD. The error is given when:

- the address of the granule to be freed is not a multiple of the number of sectors per granule
- the granule to be freed is already free
- the address of the granule to be freed is outside the DAD or zero.

The error is given by the X:ALGR task and the X:MASG task. When the X:ALGR task is involved, information on the error can be obtained from the control block given to X:ALGR (in A4).
No standard information can be given when the X:MASG task is the error causer. Only that A8 will point to an ECB, reading the GRANTB (i.e. the granule addresses to be freed) from the file to be freed.

For detailed descriptions about monitor control blocks, system error codes etc.the user should refer to Volume IV of this series, the Trouble Shooting Guide.

MANUAL COMMENT FORM

Concerns manual P800M Programmer's Guide 3 – Vol. I

12NC 5122 991 28375

including update(s) –

Originator

Name .....

Address .....

.....

.....

Comment (if possible, add a copy of the page(s) affected by the comment, marked with the proposed changes).

Please return this form to SSS Publications,
P.O. Box 245, 7300 AE Apeldoorn, The Netherlands.