AAGAARD

RCSL : 44-RT 1278
Author : H. Kold Mikkelsen
Edited : 76.04.01

PLATAUX

RC 3600 FILE SYSTEM
SYSTEM PROGRAMMER'S GUIDE

Keywords : File system, catalog, area process, cat76.

Abstract : This manual describes how to use the RC 3600 file system from Assembler programs. The user must be familiar with the MUS system.

This manual replaces the RCSL: 43-GL 741.

CONTENTS Page

1.1. Introduction

RC 3600 file system makes it possible to divide a disc drive into smaller independent units, files. These files are identified by names. The descriptions of the files, file descriptions, are kept in a catalog stored on the disc. File descriptions give among other things the name, the length, and the starting position on the disc of the file.

RC 3600 file system allows more discs to be coupled to the same RC 3600. Each disc is handled as a unit, i.e. it has its own catalog, and no one unit knows about the existence of the others. Files on different units may therefore have the same names. To know at which unit the file system has to look for a specific file it is therefore necessary to give the number of the unit besides the name of the file.

With the RC 3600 file system you can create and remove files on a disc, read and write into existing files. In addition you can change some of the information in a file descriptor, and you can look into a file descriptor.

Operations involving reading or writing the catalog are handled in a special catalog handler process, cat.

To avoid reading the catalog for each reading/writing into a file (to get the information about the file's position on the disc), you have to create an area process on the file first. When an area process is created, the file descriptor is copied to the area process description. Read/write is now carried out by sending messages to the area process.

Because the number of discs and area processes is not the same in every use of the file system, you have to define these things. This is done by making an option program containing the information needed.

The file system is able to handle all existing disc types which belong to the RC 3600 hardware system.

The file system is reentrant, i.e. many input/output operations may be processed at the same time.

1.2. The File System

When a new unit is to be used by the file system, it must be initialized. To do this, a special utility program is used (cf. 1.7.4.). Among many parameters to the utility program are two slice sizes:

a big one (> 3 segments)
a small one (> 2 segments and <big slice)

which are used when a file is to be extended.

A unit should only be initialized once, but each time it is inserted in the system the catalog must be initialized (cf. 1.4.1.).

Now the file system is able to set, create, lookup, change and remove entries, create and remove area processes, and read and write on the file.

The slices mentioned above are used in the following way:

In set, create and change entry a file size is specified.

If file size > = big slice, then the necessary number of free big slices is reserved.

If file size < big slice, then the necessary number of free small slices is reserved.

When the file does not exist, an index block is created too (cf. 1.3.3.), and the file system increments the file size by one.

When writing on a file, the reserved length may be exceeded. If so, the file system may extend the file with free small slices regardless of the slice sizes already used.

The big slices are set on the lower part of the unit, starting at the first data segment, and move towards the higher part. The small slices are set on the higher part of the unit, starting at the last data segment, and move towards the lower part.

The areas used for the two types of slices will vary during run, but will not be allowed to overlap each other. If they do so, this will cause the result (disc full) even if there are many free slices left.

1.3. Disc Format

To be able to handle arbitrary disc sizes, the file system is dealing with logical units. These may be an element of a disc pack or may cover many disc drives. These units are described in a kit description called catw (cf. 1.7.).

In the following, only the format inside the unit is described.

1.3.1. Unit Format

Segment	(0:5)	Reserved by the MUS system
	6	System index block, contains the segment numbers of the catalog file.
	7	Map index block, contains the segment numbers of the bit description.
	8	Kit description block, contains the information of the unit (cf. 1.7.).
	9	Map block, big slices.
	10	Map block, small slices.
	(11:n)	Reserved by the operating system, 'n' is set during the initialization of the unit (is typically equal to 32).
	n+1	First data segment.

The user has no access to the first (n+1) segments, but may read from the catalog file (name (<:sys:>)) and the map blocks (name (<:map:>).

The following segments are the data segments with the upper limit equal to the top-data-segment.

The first big slice is occupied by the catalog file.

1.3.2. File Format

A file consists of a number of slices which are linked together by means of an index block. The index block is always written in the first segment of the first slice, except for the catalog file (cf. 1.3.1.). The slices may thus be spread over the whole unit although the user will not see this.

1.3.3. Index Block

The file is logically contiguous by means of the index block which is written in the first segment of the first slice of the file.

Each slice of the file is described in the index block by the size and the segment address of the slice. The maximum number of slices the index block can hold is 127, which in turn gives the biggest possible file size.

The user cannot read or write the index block.

1.3.4. Catalog

The catalog is organized as an ordinary file except that the index block is written in segment number 6. The name of the catalog is <:sys:>.

Initially the first big slice is occupied by the catalog.

Each of the segments in the catalog can hold 15 entries, and each entry occupies 16 words. A name may consist of 5 characters with the 6th character equal to the unit number (cf. 1.7.). It is assigned to one of the segments by a hashkey calculation.

If the number of entries exceeds the possible 15, the catalog is extended with a big slice.

Format of an entry:

Entry	+ (0:2)	Name, 5 significant characters, the 6th = binary unit number.
	+ (3:5)	Optional initial content = 0. It is free for use by means of 'setentry' and 'lookup entry' and the content is not tested by the file system.
	+ 6	Attribute.
	+ 7	File length, during automatic extension of the file (output only), the file length indicates the actual number of segments written. Else the filelength = the reserved length.
	+ 8	Segment number of the index block, is essentially the start address of the file (cf. 1.3.3.).
	+ 9	Reserved length, the number of segments occupied by the file.
	+ (10:15)	Tail, as optional above.

Attributes: The following attributes are possible:

- Extendable, i.e. the file is extended if necessary during output.
- Entry only, i.e. only an entry in the catalog is set with the reserved length = 0.
- Writeprotected, i.e. it is only possible to read from the file.
- Permanent, i.e. the file cannot be removed or changed, yet change attribute is accepted.

1.4. Catalog Handler

All functions which involve reading/writing in the catalog are sent as messages to the process cat.

This is the case for the following functions:

1) Initializing a catalog on a disc,
3) Creating and releasing file descriptions,
4) Creating and removing area processes.

File names. - The maximum length of the file name in the file descriptor is 5 chars. In all calls of cat which involve reference to a file the file name occupies 6 chars. The first five constitute the file name, and unused positions are filled in with zero characters. The 6th character is the identification of the disc where the file has to be located. The discs in the system are numbered from zero onwards. This character is called the drive number and is stored binary.

To make it possible to distinguish between operations on entries and area processes in case of errors, bit 3 or 4 in the result is set:

1b3 : error in connection with a cat entry operation
1b4 : - - - - - area process operation

In the answers only mess0.buf = result is relevant.

1.4.1. Init Catalog

A disc must be initialized by the file handler before it is used. This is done by sending the messages stated below to the process cat.

	Message
mess0.buf	1b7
mess1.buf	not used
mess2.buf	driveno
mess3.buf	not used

The disc given by the binary number driveno in mess2 is initialized. In further use of the initialized disc this driveno. must be the 6th character in the file name.

The old catalog on the driveno. in question is used without any changes. The kit description (CATW) is read.

Result = 0, catalog initialized

Time = 3, disc accesses

1.4.2. Set Entry

To set an entry, send the message stated below to the process cat.

	Message
mess0.buf	1b0
mess1.buf	not used
mess2.buf	storage address
mess3.buf	not used

Sets an entry with the specified name and reserves the specified size. It works as 'create entry' (cf. 1.4.3.), but the file descriptor should be set by the user. In this way the optional words may be set too.

Storage address points to a 16-word area:

Address	(0 : 2)	name (6th character = binary driveno.)
	(3 : 5)	optional
	(6)	attribute
	(7)	file length (set by CAT)
	(8)	segmno of indexblk (set by CAT)
	(9)	reserved length
	(10 : 15)	tail

Attribute, see 1.4.3. Create Entry.

Result =	0,	entry set
Result <>	0,	entry is not set
	1b3+ 1b0,	catalog i/o error
	1b3+ 1b6,	parameter error, i.e.

1) wrong size, file length < 0,
2) wrong attribute, any bit to the left of 1b11 is set,
3) disc not initialized.

	1b3+ 1b7,	disc full
	1b3+ 1b11,	entry with the same name already exists
	1b3+ 1b12,	indexblk or mapblk full

Time = 16 + 4 * number of slices.
If the catalog is full, then add 10 disc accesses.

1.4.3. Create Entry

To create a new file on a disc, send the message stated below to the process cat.

	Message
mess0.buf	1b1
mess1.buf	name address
mess2.buf	size
mess3.buf	attribute

Create a new file on a disc. A file consists of a file description and some data blocks.

Name address points to a 3-word area with the name of the file. The maximum length of the name is 5 characters, and the 6th character always gives the driveno. in binary form of the disc where the file has to be created.

Attribute defines the type of the file:

1b15	:	extend possible
1b14	:	(not used)

	1b13 :	entry only, file length = 0
	1b12 :	write protected
	1b11 :	permanent file

Size (>1) gives the number of hole segments the file should occupy on the disc.

The reserved length is set to 'size' + a number of segments needed to have a whole number of slices (cf. 1.2.).

After a successful call of 'create entry', the new file description has the following content:

File descr	(0:2)	name, 6th character = binary unit No.
	(3:5)	optional = 0
	(6)	attributes
	(7)	file length - reserved length
	(8)	segmno. of indexblk
	(9)	reserved length
	(10:15)	tail = 0

Result =		0, entry created
Result <>		0, entry is not created
	1b3+	1b0, catalog i/o error
	1b3+	1b6, parameter error, i.e.

1. wrong size, file length <0,
2. wrong attribute, any bit to the left of 1b11 is set,
3. name format illegal,
4. disc not initialized.

	1b3+	1b7, disc full
	1b3+	1b11, entry with the same name already exists

Time in disc accesses = 16 + 4 * number of slices.
If the catalog is full, then add 10.

1.4.4. Look Up Entry

To look into a specific file descriptor, send the message stated below to the process cat.

	Message
mess0.buf	1b2
mess1.buf	name address
mess2.buf	entry address
mess3.buf	not used

Looks up the file descriptor given at name address (same format as in 'create entry') in the given catalog and copies the file descriptor (16 words) to entry address and onwards. For a description of file descriptor format, see 1.3.4.

Result =		0,	entry looked up
	1b3+	1b0,	catalog i/o error
	1b3+	1b1,	entry does not exist
	1b3+	1b6,	disc not initialized
Time =		2,	disc accesses

1.4.5. Change Entry

To change the file name, the file length, and the attribute for a specific file, send the message stated below to the process cat.

	Message
mess0.buf	1b3
mess1.buf	name address
mess2.buf	storage address
mess3.buf	not used

Used to change name, length, and attribute of file. All files excepted the permanented can be renamed. An area process must not exist on the file.

Name address points to a 3-word area with the name of the file in the 5 first characters, and the binary driveno. in the 6th character.

Storage address points to an area with description of the new content:

Address	(0 : 2)	new name, 6th character = binary unit No.
	(6)	new attribute
	(9)	new file length

New attribute : If new attribute <0, then it is not changed.

New file length : If new file length <0, then it is not changed.

New name : If address + 0 = 0, the name is not changed.
If the file is permanented, the name is not changed, and the result (illegal) is set.

Result =

	0, entry changed
1b3+	1b0, catalog i/o error
1b3+	1b1, entry does not exist
1b3+	1b7, not enough disc space for file.
1b3+	1b6, illegal

1. name format illegal,
2. disc not initialized,
3. change of name for a permanent file,
4. wrong attribute, any bit to the left of 1b11 is set,
5. an area process exists on the file.

1b3+	1b11, entry with new name already exists

Time in disc accesses =

a :	change attribute :	3
b :	change name :	8
	If the catalog is full, then add	10
c :	change length :	
	If new length > old length, then	8
	If old length = 0, then add	3
	If new length < old length, then	4 + 2 * number of slices

Combinations: a + b = b; a + c = c; b + c = b + c; a + b + c = b + c.

1.4.6. Remove Entry

To remove a file from a disc, send the message stated below to the process cat.

	Message
mess0.buf	1b4
mess1.buf	name address
mess2.buf	not used
mess3.buf	not used

If an area process exists on that file, the message is rejected, else the entry is deleted.

Name address points to a 3-word area with the name of the file in the 5 first characters, and the binary driveno. in the 6th character.

Result =	0,	entry removed
1b3+	1b0,	catalog i/o error
1b3+	1b1,	entry does not exist
1b3+	1b6,	illegal, i.e.

1. name format illegal,
2. disc not initialized,
3. area process exists on the file.

Time in disc accesses = 8 + 2 * number of slices

1.4.7. Create Area Process

Before reading and writing a disc file, it is necessary to create a process to which you can send the read, write, and control messages. This is done by sending the message described below to the process catalog.

	Message
mess0.buf	1b5
mess1.buf	name address
mess2.buf	not used
mess3.buf	not used

Makes a file on a given disc available for the calling process as an area process. The name of the area process is the 5-character name of the file descriptor extended by the binary driveno. If the area process does not exist, the file descriptor is looked up in the catalog on the given disc, and an area process descriptor is taken from the queue of free area process descriptors and initialized according to file description. It is chained to processchain and started as a normal MUS process.

If the area process already exists, the message is dummy.

Name address points to a 3-word area with the name of the file as the first 5 characters, and the binary driveno. as the 6th character.

Result =	0,	area process created or already exists
1b4+	1b0,	catalog i/o error
1b4+	1b1,	entry does not exist
1b4+	1b6,	illegal, i.e. 1. name format illegal, 2. disc not initialized.
1b4+	1b7,	no area process descriptor available

Time in disc accesses =	area process exists:	0
	area process does not exist:	2

1.4.8. Remove Area Process

When finishing the reading and writing a disc file, you can release the used area process by sending the message stated below to the process catalog.

	Message
mess0.buf	1b6
mess1.buf	name address
mess2.buf	not used
mess3.buf	not used

If any user of the area process exists or the area process does not exist, the message is dummy, else

- all messages in the eventqueue are returned,
- the area process is removed,
- the catalog on the disc is updated.

Name address points to a 3-word area with the name of the file as the first 5 characters, and the binary driveno. as the 6th character.

Result = 0, process removed or usercount > 0 or process does not exist.

1b4+ 1b0, catalog i/o error

1b4+ 1b6, illegal, i.e.

1. name format illegal
2. disc not initialized

Time in disc accesses = process does not exist: 0

usercount > 0 : 0

usercount = 0 : 3

1.5. Area Processes

A description of how to create and remove files on disc, initialize catalogs, and create and remove area processes is to be found in chapter 1.4.

The name of an area process consists of 6 characters. The first 5 correspond to the name of the disc file, and the 6th 'character' is the number of the disc on which the file is stored. This number should be binary.

The first block in a file has the relative number zero.
Both control and transput messages are accepted.

1.5.1. Control

Reservation for use, exclusive user, and exclusive writer are accepted. Besides these only position is accepted.

Reservation: If it is possible, the calling process may be included as user, exclusive user, or exclusive writer, or the user may be removed.

At the most 3 different users are possible, and each user has an open/close count which follows the number of reservation commands to the user entry.

Count = 2 : Insert user as exclusive writer.
All other processes are allowed to read the file.
The open/close count is increased.
If the user is new, the usercount is increased.

Result = 0 : The reservation is accepted.
1b6 : Area process reserved for exclusive use by another process.
1b4 + 1b6 : The area process is already reserved for exclusive writing by another process.
1b4 + 1b12 : No more user entries.

Count = 3 : Insert user.

All other processes are allowed to read the file, and the area process may be reserved for exclusive writing too.

The open/close count is increased.

If the user is new, the usercount is increased.

Result = 0 : The user is inserted as a user of the area process.

1b6 : The area process is reserved for exclusive use by another process.

1b4 + 1b12 : No more user entries.

Count = 4 : Insert user as exclusive user.

Any other process is rejected.

The open/close count is increased.

The usercount can only be 1.

Result = 0 : Reservation accepted.

1b6 : Area process reserved for exclusive use by another process.

1b4 + 1b6 : Area process is being used by another process.

1b4 + 1b12 : No user entry.

Count = 0 : Remove user.

The open/close count is decreased. If it reaches zero, the usercount is decreased too, and if the user was either exclusive user or writer, this state is cleared.

Result = 0 : The user is removed.

1b6 : Area process reserved for exclusive use by another process.

1b4 + 1b11 : The calling process is not a user of the area process.

Time = 0 disc accesses

Position: The position in mess3.buf within the file is set.

If the position < 0, then the position : = 0.
If the position > file size, then the position : = file size.

Result = 0 : Position accepted.

1b6 : Area process reserved for exclusive use by another process.

1b4 + 1b6 : Position < 0.

1b4 + 1b11 : 1. Position is outside the file.
2. The sender is not a user of the area process.

Time = 0 disc accesses.

Other control messages: The disc is sensed to check the status.

Result = 0 : OK

1b6 : Area process reserved for exclusive use by another process.

1b4 + 1b11 : The sender is not a user of the area process.

Time = 0 disc accesses.

1.5.2. Transput

	Message	Answer
mess0.buf	operation	status
mess1.buf	bytecount	number of bytes transferred
mess2.buf	byteaddress	unchanged
mess3.buf	(first block No.)	first block No.

Bytecount in mess1.buf must be 512 bytes.
Byteaddress in mess2.buf must be even.

Input

operation = 1 Input sequential mode. Reading of the file continues at the position reached by the last transput or position message.

Mess3.buf is irrelevant in the message, but in the answer it is set to the number of the block first read.

operation = 5 Input random mode. Before reading the file is positioned to the position given in mess3.buf. Else as for operation = 1.

Output

operation = 3 Output sequential mode. Writing in the file continues at the position reached by the last transput or position message.

The file is automatically extended if attribute (extension possible) is set for the entry at an amount corresponding to the slice size chosen (cf.

Mess3.buf is irrelevant in the message but in the answer it is set to the number of the block first written in answer.

operation = 7 Output random mode. Before writing the file is positioned to the position given in mess3.buf. Else as for operation = 3.

operation = 19 Read after write in sequential mode. First an output operation = 3 is carried out. Then the written segment is read for check of status. For some disc devices the output buffer is used as input buffer during the check read.

operation = 23 Read after write in random mode. Works as operation = 19, except that output operation = 7 is carried out.

Time in disc access:

Input : If position is in the position entries in the area process, then 1, else 2.

Output : If position is in the position entries in the area process, then 1, else

if position is inside the reserved file, then 2, else 10.

If read after write in mode, then add 1.

BIT	CAT76 + 1b3 or + 1b4 *)	RC 3652 - 2, 4 Mb	RC 3650	RC 8221, 22, 23, 24, 25, 26
1b0	Catalog i/o error	Unit not available, malfunction (hardware (12)).	Disconnected, unable to accept commands.	Disconnected, power off, hard error, unit not available.
1b1	Entry does not exist	-	Off line	-
1b2	-	-	-	-
1b3	Message from CATHANDLER	-	-	-
1b4	Message from area process	-	-	-
1b5	-	-	Write-protected	Write-protected
1b6	Name format illegal, disc not initialized, attribute error, file length <0 , area process reserved. (1b6 - only).	Driver reserved by another process	Driver reserved by another process	Driver reserved by another process
1b5+1b6	-	Unit does not exist	Write-protected in output	Unit does not exist, multiple or no unit selected, invalid drive command, write-protected in output.
1b7	Disc full (+1b3), no more area processes.	-	-	-
1b8	-	Block error	Block error	Block error
1b6+1b8	-	Odd byteaddress	-	Odd byteaddress
1b9	-	Data late	-	-
1b10	-	Parity error	Parity error	Parity error
1b11	Entry exists (+1b3), not user of area process (+1b4), position is outside the file (+1b4).	-	End of existing kit	End of existing disc kits
1b12	Map or index block full, no more user entries (+1b4).	Seek error (position error)	Hardware position error, outside existing kit (+1b11).	Hardware position error
1b13	-	-	-	-
1b14	-	Time-out	Time-out	Time-out
1b15	-	-	-	-

*) If nothing else is mentioned, one of these bits is added to the error bits in this column.

1.6. Format for Area Process Description

An area process description occupies 74 words, the first 25 correspond to a normal driver process description (including reser.proc.), see MUS Programmer's Guide, Part 2, 2.4-2.5. The remainder of the area process description is:

address.proc.	+25	attribute
	+26	file length (= actual length $\leq$ reserved length)
	+27	segment number of start of first slice (= address of the index block)
	+28	reserved length (= the sum of all the reserved segments)
	+(29:42)	work locations (to make the catalog systems re-entrant, for internal use only)
; user positions:		
	+43	next position entry (= next victim when a position is not found among the current user positions)
	+(44:46)	user position 1. Contents: +44 segment number of the slice +45 first block number rel. to file start +46 top block number rel. to file start
	+(47:49)	user position 2. Contents as for position 1.
	+(50:52)	user position 3. Contents as for position 1.
; user entries:		
	+53	exclusive writer
	+54	user count (see 1.5)
	+(55:57)	user entry 1. Contents: +55 user identification (= process descriptor address of user) +56 user position rel. to file start (after a transput, mess3.buf : = user position) +57 open/close count (see 1.5)

+(58:60) user entry 2. Contents as for user entry 1.

+(61:63) user entry 3. Contents as for user entry 1.

; message buffer:

+(64:73) message buffer (see MUS Programmer's Guide, Part 2, 2.6)

1.7. Loading the File System

The file system consists of the following sections:

1. The file handler program, named cat.
2. An option program, named catw.
3. An option program, named capx, where x = number of area processes (e.g. cap8).
4. One or more disc drivers; their names are given in catw.

When all these programs exist as binary tapes and the MUS system is loaded in RC 3600, you load them into RC 3600 by means of the S-command LOAD.

1.7.1. Producing catw

Each RC 3600 installation must make its own option program catw, which

1. gives the configuration of the discs connected to RC 3600;
2. contains the number of area process descriptions the system will use simultaneously.

The option program is translated with a MUS assembler to produce the binary tape. The format for catw is given below together with an example.

Format of catw

Start word

+	0	program specification (= 1b0 + 1b1 + 1b7 + 1)
+	1	start of program
+	2	program chain (set by the MUS system)
+	3	size of program
+	(4:6)	name of program (= < : catw <0> : >)
+	7	first area process
+	8	top area process = first area process

; head of unit chain:

+	9	head of unit chain
+	10	chain of head of unit chain

; unit chain (a):

+ 11	chain of unit < a >, points to the next unit in the chain
+ 12	size of unit < a > descriptor (= 18)
+(13:15)	name of unit : < : unit < a > < 0 > : >
+(16:18)	name of driver which handles the unit
+(19:20)	kit displacement = number of segments prior to unit < a >, which is handled by the driver.

; unit parameters:

+(21:28) additional descriptor

+21 normal slice size (= big slice size), constant set by user: 2 < size < 256.

+22 increment slice size (= small slice size), constant set by user: 1 < size < normal.

+23 number of segments on unit < a >, constant set by user.

+24 number of free segments on unit < a >, init. value = number of data segments = number of segments on unit <a> - first data segment, set by user, changed by cat during run.

+25 first data segment, points to the catalog file for unit < a >, constant set by user.

+26 top data segment, points to the last segment +1 of unit <a>, = number of segments on unit < a >, constant set by user.

+27 min. slice, the last reserved normal slice number and is used to calculate the limit for extension with increment slices, init. value : = 0 by the user, changed by cat during run.

+28 max. slice, the last reserved increment slice number and is used to calculate the limit for extension with normal slices, init. value : = 0 by the user, changed by cat during run.

; unit chain (b):

+(29:46) contents as for unit chain (a) except that the unit number must be <b>.

.

.

.

.

; unit chain (n), the last unit in the chain:

last.addr+ 0 chain of unit <n> : = 0, i.e. no more units in the chain.

last.addr+ (1:17) contents as for unit chain <a> (12:28) except that the unit number must be <n>.

The unit chain is followed by:

; 1 area process:

; process descriptor: The next 22 words contain the normal MUS process description (including save.cur). The name is <: catw <0>:>, and the priority is -1.

; start address of catw program (word (+1)):
A piece of code which inserts the catw-process and the code as an area process in the area process chain, which means in turn that the catw-process is removed (the catw-program is still in the program chain).

. blk <expression>

where <expression> + size of (catw-process descriptor)
+ size of (catw-program code)
= size of area process = 74

An example of a disc description is given in the following pages. It contains two disc units and one flexible disc.

Please note that the unit descriptors are structured as programs, but neither the specification nor the program start makes any sense, and for this reason those two first words are omitted from the unit chain, but if the basis system should be able to handle the chains, the chain pointer must take into account the lack of the first two words. This is the reason for the "-2" in the chain pointers.

```
; rc3600 disc catalog system cat76, disc description.

.titl catw
.nrel
.txtm
.rdx 10

dw0:                            ; program (catw)
        lb0 + lb1 + 1           ;     specification
        dw4                     ;     program start
        0                       ;     program chain
        dw5 - dw0               ;     size of program
        .txt  .catw <0>.        ;     name of program

dw1:    darea                   ; first area process
        darea                   ; top area process

; head of unit chain:
        dhead -2                ; head of unit chain
dhead:  dk0 -2                  ; chain of head of unit chain

; chain of units:
  dk0:  dk4 -2                  ; unit 0: chain of unit 0
        dk4 -dk0                ;     size of unit 0 descriptor
        .txt .unit < 0>.        ;     name of unit 0
        .txt .fd0 <0> <0>.      ;     name of rc 3650 disc driver
        0                       ;     kit displacement (0:15)
        0                       ;     kit displacement (16:31)
        .blk 8                  ;     unit 0 parameters
```

```
dk4:    dk4 -2              ; unit 4: chain of unit 4
        dk5 -dk4            ;    size of unit 4
        .txt .unit <4>.     ;    name of unit 4
        .txt .dkp0 <0>.     ;    name of rc 3652 disc driver
        0                   ;    kit displacement (0:15)
        0                   ;    kit displacement (16:31)
        .blk 8              ;    unit 4 parameters

dk5:    0                   ; unit 5: chain of unit 5, last unit
        darea -dk5          ;    size of unit 5
        .txt .unit <5>.     ;    name of unit 5
        .txt .dkp0 <0>.     ;    name of rc 3652 disc driver
        0                   ;    kit displacement (0:15)
        9744                ;    kit displacement (16:31)
        .blk 8              ;    unit 5 parameters
```

```
; 1 area process:
  darea:
; process descriptor:        (used as area process later on)
  dw5:                       ; process (catw)
        0                    ;     next process
        0                    ;     prev. process
        0                    ;     chain
        dw6 - dw5            ;     size of process
        .txt .catw < 0>.     ;     name of process
        dw5 + event          ;     first event
        dw5 + event          ;     last event
        0                    ;     buffer
        dw0                  ;     program
        0                    ;     state
        0                    ;     timer
        -1                   ;     priority
        dw4                  ;     break address
        dw4                  ;     ac0
        0                    ;     ac1
        dw4                  ;     ac2
        0                    ;     ac3
        dw4*2                ;     psw
        0                    ;     save

; start address of catw:
  dw4:  lda 2 cur            ;
        lda 0 process        ;
        lda 1 areap          ;
        rechain              ;
        lda 2 cur            ;
        stopprocess          ;
  dw7:                       ;
        .blk 74-dw7+dw5      ;
  dw6:                       ; top of process (catw)
        .end dw5             ;
```

1.7.2. Loading capx

If the user wants more than 1 area process (given by catw), he may either link free core to the area-process chain or use the utility program capx, where x = number of area processes to be added.

Every time the capx is loaded by the MUS system, x area processes are linked to the existing area processes in the area process chain. The core occupied by the capx is used as area process area and the capx process is removed. Only the first 7 words remain in the core.

By combining different capx-programs the user may have exactly the number of area processes he wants.

1.7.3. Initnew Catalog

When a new disc unit is to be used, it must be initialized. To do this, the following output messages are sent to the process cati.

	Message
mess0.buf	1b8 + 3
mess1.buf	bytecount = 18
mess2.buf	byteaddress
mess3.buf	not used

The byte address must point to an 18-byte record with the content:

wordaddress = byteaddress

+0	Unit number of the unit to be initialized
+(1:8)	Unit parameters as described in 1.7.1., start word +(21:28).

It should be noted that the unit catalog and the maps are cleared regardless of the original content.

The process cati will only accept the commands: reservation and output with 1b8. The commands position, cathandler, input and output without 1b8 are illegal. All other commands will only sense the disc driver.

Result =	0 :	Unit initialized
	1b6 :	The process is reserved by another user
	1b3 + 1b6 :	The kit or unit description is not found (catw)
	1b4 + 1b6 :	Illegal command 1. Catmessage received (cf. 1.4.) 2. Position command received 3. Output message but not initnew
	1b3 + 1b8 :	Bytecount <> 18
Time =		17 disc accesses

1.7.4. Unit Numbers

The binary unit numbers in the names and in catw are free for use within the limits (0-255). In catw the connection between the unit number and the physical disc is made by the driver name and the kit displacement.

It is recommended to use the unit start number corresponding to the different type of discs as shown below. The main idea is for the user of a given configuration to start with the unit numbers listed.

For some special applications the user might have many catalogs (and thus units) on each physical disc kit, and the unit numbers will exceed the start number of the next disc type in the list. But in these cases the next disc type is implicitly excluded and the corresponding unit numbers are free.

Disc Type	Size	Start Unit No.
RC 3650	1/4 Mbyte	0
RC 3652	2.5 Mbytes	4
RC 8221	12 -	8
RC 8222	18 -	16
RC 8223	33 -	32
RC 8224	66 -	48
RC 8225	124 -	64
RC 8226	248 -	128