issn 0065-3713

INSTITUT D'AERONOMIE SPATIALE DE BELGIQUE

3 · Avenue Circulaire
B · 1180 BRUXELLES

AERONOMICA ACTA

B – N° 65 – 1996

MACSIMS

Handleiding voor de bediening van het instrument

Commando's, telemetingpakketten, *scripts*, scenario en controlelijst uit de vlucht van 23 november 1995

door

E. Neefs

BELGISCH INSTITUUT VOOR RUIMTE-AERONOMIE

3 · Ringlaan
B · 1180 BRUSSEL

Voorwoord

In dit document werden de commando's, de telemetingpakketten, de scripts, het scenario en de controlelijst voor het MACSIMS-instrument gebundeld (versie van de ballonvlucht van 23 november 1995).

Avant-propos

Dans ce document les commandes, les paquêts de télémesure, les scripts, le scénario et la liste de contrôle pour l'instrument MACSIMS (version du vol ballon du 23 novembre 1995) ont été rassemblés.

Foreword

This document bundles the commands, the telemetry packets, the scripts, the scenario and the control list for the MACSIMS instrument (version of the flight of November 23th, 1995).

Vorwort

In dieses Dokument sind die Kommandos, die Telemetriepakete, die Scripts, das Scenario und die Kontroleliste für das MACSIMS Instrument gebündelt (Version des Ballonfluges von 23 November 1995).

Commando's

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commandlist
# version : flight 1995
#
############################################################
```

LIST OF EXISTING COMMANDS on BOTH PROCESSORS
****

1. Global variables

```
GV=STRING (ex. PROMPT=STRING changes the system prompt automatically)
GV=VALUE
set
unset
unset GV (ex. unset PROMPT restores default system prompt)
let GV = VALUE+VALUE
let GV = VALUE-VALUE
let GVi = $GVj+VALUE
let GVi = $GVj-VALUE
let GVi = VALUE+$GVj
let GVi = VALUE-$GVj
let GVi = $GVj+$GVk
let GVi = $GVj-$GVk
```

2. Editing and executing scripts

```
cat SCRIPTNAME  cat > SCRIPTNAME

ctrl_C (buttons 'CNTRL' and 'C')

SCRIPTNAME

SCRIPTNAME [PAR1 [PAR2 ... [PAR9]]]]]]]]]
               positional parameter PAR1 corresponds in script with $1
               positional parameter PAR2 corresponds in script with $2
                                  ....
               positional parameter PAR9 corresponds in script with $9
```

3. Script and directory maintenance

```
cp SCRIPTNAME1 SCRIPTNAME2
```



```
dir

fsck

rm SCRIPTNAME
```

4. Background processing

```
(any command) &

kill -i   kill TASK_ID
kill $GV1
```

5. Special commands

```
sleep TIME
sleep $GV1

echo ARGUMENT1 ARGUMENT2 ... ARGUMENTn
echo $GV1

date
date DDD HH MM SS
date $GV (GV=DDD HH MM SS)
date $GV1 $GV2 $GV3 $GV4

check
```

6. Loops in scripts

```
for VAR in L1 [L2 [L3 ... [L9]]]]]]]]
do
...
done

while test $GVi COMP VALUE
do
...
done

while test VALUE COMP $GVj
do
...
done

while test $GVi COMP $GVj
do
...
done
```

7. Secure

To avoid modifications of data resulting in a definitive blocking of the processor, the "secure" command protects critical RAM-segments, such as the code-segment, the constants-segment, the segment with calibration tables for all types of voltages and the script directory segment.

A delibarate modification in one of these segments has to be preceeded by the execution of the "insecure" command. After modification the "secure" command has to be run again.

8. Remarks

- The date commands with an argument (i.e. the commands that set the date) in the MOBC set simultaneously the date in the CFMP. Therefore both processors should be switched on to execute a setting date command from the MOBC command line. The date in the CFMP can be set independent from the MOBC (remsh date with arguments).
- It is preferable not to use $1,...,$9 in commands from the command line. They are reserved for use in scripts.
- loops can only be used in scripts
- the rm, cp and cat-commands should not be used in scripts (safety)

LIST OF EXISTING COMMANDS on MOBC ONLY

1. High Voltage Control

```
hvgen on    hvgen off
hvgen $GV1

cpv    cpv VALUE    cpv on    cpv off    cpv -t    cpv + VALUE    cpv -r VALUE
cpv -m VALUE1 VALUE2    cpv -M VALUE1 VALUE2    cpv -pa    cpv -px
cpv $GV1    cpv $GV1 $GV2    cpv $GV1 $GV2 $GV3

itv    itv VALUE    itv on    itv off    itv -t    itv + VALUE    itv -r VALUE
itv -m VALUE1 VALUE2    itv -M VALUE1 VALUE2    itv -pa    itv -px
itv $GV1    itv $GV1 $GV2    itv $GV1 $GV2 $GV3


pav    pav VALUE    pav on    pav off    pav -t    pav + VALUE    pav -r VALUE
pav -m VALUE1 VALUE2    pav -M VALUE1 VALUE2    pav -pa    pav -px
pav $GV1    pav $GV1 $GV2    pav $GV1 $GV2 $GV3

psv    psv VALUE    psv on    psv off    psv -t    psv + VALUE    psv -r VALUE
psv -m VALUE1 VALUE2    psv -M VALUE1 VALUE2    psv -pa    psv -px
psv $GV1    psv $GV1 $GV2    psv $GV1 $GV2 $GV3
```


2. RF/DC Control

```
grid    aux VALUE    aux on    aux off    aux -t    aux + VALUE    aux -r VALUE
grid -m VALUE1 VALUE2    aux -M VALUE1 VALUE2    aux -pa    aux -px
grid $GV1    aux $GV1 $GV2    aux $GV1 $GV2 $GV3

iiv    iiv VALUE    iiv on    iiv off    iiv -t    iiv + VALUE    iiv -r VALUE
iiv -m VALUE1 VALUE2    iiv -M VALUE1 VALUE2    iiv -pa    iiv -px
iiv $GV1    iiv $GV1 $GV2    iiv $GV1 $GV2 $GV3

rf    rf VALUE    rf on    rf off    rf -t    rf + VALUE    rf -r VALUE
rf -m VALUE1 VALUE2    rf -M VALUE1 VALUE2    rf -pa    rf -px
rf $GV1    rf $GV1 $GV2    rf $GV1 $GV2 $GV3

opb    opb VALUE    opb on    opb off    opb -t    opb + VALUE    opb -r VALUE
opb -m VALUE1 VALUE2    opb -M VALUE1 VALUE2    opb -pa    opb -px
opb $GV1    opb $GV1 $GV2    opb $GV1 $GV2 $GV3
```

3. Communication with T4 processor

```
msp VALUE    msp -s    msp -l VALUE    msp -t VALUE    msp -b VALUE
msp $GV1    msp $GV1 $GV2
```

4. Housekeeping, valve and shutter commands

```
thk      thk -tm      ahk      ahk -tm
thk $GV1              ahk $GV1
valve open     valve close     valve status
valve $GV1
cap      cap test      cap enable      cap confirm      cap open
cap disable
cap $GV1
mks100 on      mks100 off
mks100 $GV1
gauge on      gauge off
gauge $GV1
```

5. Time of flight commands

```
tof -s  tof VALUE        tof -r VALUE     tof -d VALUE     tof -p VALUE
tof -g VALUE1 VALUE2
```

LIST OF EXISTING COMMANDS on CFMP ONLY

All these commands are given during flight from the MOBC command line. They should be preceeded by the keyword "remsh".

1. Turbo Pump Control

```
turbof   turbof VALUE   turbof on   turbof off   turbof -t   turbof + VALUE
turbof -r VALUE   turbof -m VALUE1 VALUE2   turbof -M VALUE1 VALUE2
turbof -pa   turbof -px   turbof $GV1   turbof $GV1 $GV2
turbof $GV1 $GV2 $GV3

turboi   turboi VALUE   turboi on   turboi off   turboi -t   turboi + VALUE
turboi -r VALUE   turboi -m VALUE1 VALUE2   turboi -M VALUE1 VALUE2
turboi -pa   turboi -px   turboi $GV1   turboi $GV1 $GV2
turboi $GV1 $GV2 $GV3
```

2. Housekeeping

```
chk     chk -tm
chk $GV1
```

3. Cl Control

```
valve Cl in     valve Cl out    valve Cl close  valve Cl status

flow Cl   flow Cl on    flow Cl off     flow Cl -t      flow Cl + VALUE
flow Cl -r VALUE   flow Cl -m VALUE1 VALUE2   flow Cl -M VALUE1 VALUE2
flow Cl -pa   flow Cl -px   flow Cl $GV1   flow Cl $GV1 $GV2
flow Cl $GV1 $GV2 $GV3

press Cl        press Cl on     press Cl off
```

4. I Control

```
valve I in      valve I out     valve I close   valve I status

flow I   flow I on      flow I off      flow I -t       flow I + VALUE
flow I -r VALUE   flow I -m VALUE1 VALUE2   flow I -M VALUE1 VALUE2
flow I -pa   flow I -px   flow I $GV1   flow I $GV1 $GV2
flow I $GV1 $GV2 $GV3

press I         press I on      press I off
```

5. Flush Control

```
valve X open    valve X close   valve X status

flow X on       flow X off
```



```
flow XCl   flow XCl on  flow XCl off    flow XCl -t     flow XCl + VALUE
flow XCl -r VALUE   flow XCl -m VALUE1 VALUE2   flow XCl -M VALUE1 VALUE2
flow XCl -pa   flow XCl -px   flow XCl $GV1   flow XCl $GV1 $GV2
flow XCl $GV1 $GV2 $GV3

flow XI   flow XI on    flow XI off     flow XI -t      flow XI + VALUE
flow XI -r VALUE   flow XI -m VALUE1 VALUE2   flow XI -M VALUE1 VALUE2
flow XI -pa   flow XI -px   flow XI $GV1   flow XI $GV1 $GV2
flow XI $GV1 $GV2 $GV3

press X         press X on      press X off
```

6. Discharge Control

```
dis   dis VALUE   dis on   dis off  dis pulse  dis -t   dis + VALUE
dis -r VALUE   dis -m VALUE1 VALUE2   dis -M VALUE1 VALUE2
dis -pa   dis -px   dis $GV1   dis $GV1 $GV2
dis $GV1 $GV2 $GV3
```

7. Lamp Control

```
lamp   lamp VALUE   lamp on   lamp off  lamp pulse  lamp -t   lamp + VALUE
lamp -r VALUE   lamp -m VALUE1 VALUE2   lamp -M VALUE1 VALUE2
lamp -pa   lamp -px   lamp $GV1   lamp $GV1 $GV2
lamp $GV1 $GV2 $GV3
```

8. Special commands

```
remote          local
```

```
#############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/summary
# version : flight 1995
#
#############################################################
```

```
************************************************
*                                              *
*  M A C S I M S     M A N U A L   P A G E S   *
*                                              *
*               FLIGHT 1995                    *
*                                              *
************************************************
```

SUMMARY

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/cat
# version : flight 1995
#
############################################################
```

NAME

cat - fill or list script

SYNOPSIS

```
cat    scriptname
cat >  scriptname
```

DESCRIPTION

```
cat    scriptname       lists the content of a script
cat >  scriptname       copy lines from standard input to script
```

The scriptname may contain upto 15 printable characters.

Standard input is the TeleCommand channel (TC).

During copy or append from standard input each line terminated by a new line (line feed) is added to the script. The cat command is then terminated by entering a last line containing only a "." .

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/ctrl_C
# version : flight 1995
#
############################################################
```

NAME

ctrl_C - kills a command or script that runs in foreground

SYNOPSIS

press the 'CONTROL' and the 'C' button at the same time

DESCRIPTION

This option provides the possibility to halt commands or scripts started up in foreground.

It can be very useful when a mistake has been made in a command or script that will take a long time to execute. It can be stopped instantaneously by pressing 'ctrl_C'.

EXAMPLES

A command 'sleep 30000' has been entered instead of 'sleep 3000'. Rather than to wait 300 sec. you can halt execution immediately by pressing 'ctrl_C'.

In a script a loop has been defined of 100 cycles instead of 10 cycles. 'ctrl_C' ends the execution of the script prematurely.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/date
# version : flight 1995
#
############################################################
```

NAME

date - gets or sets the day and the time.

SYNOPSIS

```
date
date DDD HH MM SS
```

DESCRIPTION

date	gets the current day and time and the time elapsed since the last start up
date DDD HH MM SS	sets the day and time

'date' gets the day in the year [DDD], the local time in the day (hours [HH], minutes [MM], seconds [SS]) and the time that has elapsed since the last set up of the processor (hours [HH], minutes [MM], seconds [SS], ticks of 10 ms).

```
1 <=  DDD  <= 365 (366)
0 <=   HH  <= 23
0 <=   MM  <= 59
0 <=   SS  <= 59
0 <= ticks <= 99
```

'date' retuns a warning message when the day and time has not been set since last setting up the microprocessor.

'date DDD HH MM SS' sets the current day to DDD and the current time to HH hours, MM minutes, SS seconds.

'date DDD HH MM SS' returns an error message when DDD, HH, MM or SS exceeds the prescribed boundaries.

EXAMPLES

date — gets the current day and time, and also the time elapsed since the last start up, e.g.:


```
Daynumber .............: 303
Real time .............: 9h 18m 12s
Time since start up ..: 0h 14m 55s 78ticks
```

date 144 15 23 48 — sets the current day to 144 and the current time to 15h 23m 48s

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/dir
# version : flight 1995
#
############################################################
```

NAME

dir - list contents of script directory on standard output

SYNOPSIS

dir

DESCRIPTION

dir lists the contents of the script directory.

The list format is as follows:

- title: Directory of scripts
- for each script: scriptname + scriptlength in bytes
- number of scripts stored
- number of bytes occupied
- number of free bytes

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/echo
# version : flight 1995
#
############################################################
```

NAME

echo - echo arguments to standard output

SYNOPSIS

echo [arg] ...

DESCRIPTION

echo writes its arguments separated by blanks and terminated by a new-line on the standard output.

echo is useful for producing diagnostics in command scripts.

Standard output is the Telecommand echo telemetry channel.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/forloop
# version : flight 1995
#
############################################################
```

NAME

for-loops in scripts

SYNOPSIS

```
for VAR in LIST
do
command 1
command 2
...
command n
done
```

DESCRIPTION

When a script has been opened for editing (see manual page 'cat') it is possible to introduce a for-loop, enabling repeated executions of commands using a common parameter VAR. The loop consists of an 'opening' line and the loop body.
The first line of the loop body is always "do", the last line "done".

The opening line contains the name of a parameter (VAR) which will be temporarily a global variable. This global variable will be deleted after the last iteration of the loop.
The opening line contains further more a list (LIST) of values. The list has to contain at least one value and can contain maximum 10 values. VAR will then take for each iteration the next value from LIST and apply this value everywhere in the body of the loop where VAR is mentioned.

Nesting of loops (for-loops in for-loops or while- and for-loops) is possible, but not indefinitely. It is advised not to nest more than four levels.
Be careful! Also calling a script from another script is one level off nesting!!!

EXAMPLES

Imagine a global variable TRUE with the value "*****"
Imagine a script "example" that will be called with one positional

parameter : "example nice".
The script contains the following lines :

```
for TEST in text 100 Betty $TRUE $1
do
echo $TEST
done
```

Executing the script will output :

```
text
100
Betty
*****
nice
```

During execution of the for-loop a second global variable will be created in the global variable list : TEST. It is necessary that the list of global variables in not full yet (max. 10 variables).

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/fsck
# version : flight 1995
#
############################################################
```

NAME

fsck - cleans up the directory.

SYNOPSIS

fsck

DESCRIPTION

fsck cleans up the directory

fsck resets the script index (this is the counter that holds the number of existing scripts) and the total length (this is the counter that holds the number of existing characters in all existing scripts) to 0.

Note that script index has a maximum : SCRIPT_TBL_SIZ.
Note that total length has a maximum : SCRIPT_FIL_SIZ.

Additionally the "directory structure" :

```
struct { char name[FIELD_LENGTH];       /* script name */
         char *thisscript;              /* pointer to this script */
         char *nextscript;              /* pointer to next script */
}
```

is reset. I.e. the directory name is erased, and the pointers to this and the next script are redirected to the beginning of the directory.

After completion the message "Directory cleaned" will appear, followed by an empty directory list.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/glob_vars
# version : flight 1995
#
############################################################
```

SUBJECT

global variables

DESCRIPTION

Every global variable has a NAME and a VALUE.
A global variable is referenced by its name, preceeded by the dollar sign ($NAME).

A global variable is assigned a value as follows :
NAME=VALUE

The VALUE of a global variable can be of any type. Of course using a global variable with a string value in a command expecting an integer or real number will result in an error.

Maximum 10 global variables can be created.

A list of global variables can be obtained using the command "set".

A global variable can be purged from the list using :
unset NAME

All global variables can be deleted using the command "unset".

The value of the global variable can be changed by redefining the variable. When the VALUE is an integer it can be given an other value with the "let"-command.

Using global variables the possibility exists to write scripts in a general way. Without global variables the smallest change of command parameters inside a script would result in rewriting the complete script.
Now, with global variables, we can use a global variable as parameter for a command. In other words, the NAME of the global variable appears in the command, but the VALUE can be changed at any time. Consequently one script can be used for different values of the command parameter represented by the global variable.

Most of the commands have one or more parameters. It is possible

to replace one or more of these parameters by global variables, as well when the commands are issued from the command line as when they appear in scripts. This is the case for the commands : sleep, echo, kill, valve, mks10, mks100, gauge, cap, msp, hvgen, itv, cpv, psv, pav, rf, iiv, aux, opb, ahk, thk and
This is also true for the command "date" (one or more of the four parameters day, hour, minutes, seconds can be replaced by global variables). With the command "date" the possibility exists to use one global variable containing the complete date (four parameters).
Exceptions to this rule are the commands "cat", "cp" and "rm". It is safer to execute these commands from the command line and with the explicit script names, in order to avoid unwanted changes in the scripts directory.

EXAMPLES

Creating global variables :

```
EXAMPLE=nice
TIME=1200
```

Referencing global variables :

```
echo $EXAMPLE   ----> will print "nice"
sleep $TIME     ----> the processor will wait
                      12 seconds before returning
                      the system prompt
```

LIST OF POSSIBILITIES

$GV means : a global variable can be used on this spot instead of a value.

```
SCRIPTNAME $GV [$GV ... [$GV]]]]]]]]
sleep $GV
echo $GV
kill $GV
date $GV $GV $GV $GV  or  date $GV with GV=DDD HH MM SS
valve $GV
mks10 $GV
mks100 $GV
gauge $GV
cap $GV
msp $GV [$GV]
hvgen $GV
itv     |
pav     |
psv     |
cpv     |   $GV [$GV [$GV]]
rf      |
iiv     |
aux     |
opb     |
```

```
ahk $GV
thk $GV
```

REMARKS

* A special global variable is PROMPT.
  By default no global variable PROMPT exists and a
  predefined string is used as system prompt.
  Creating PROMPT (PROMPT=....) automatically changes
  the system prompt to the VALUE of the global variable
  PROMPT. E.g. "PROMPT=Test" will give "Test>" as a system
  prompt.
  The command "unset PROMPT" restores automatically the
  default system prompt.

* A manual page exists for the commands "set", "unset"
  and "let".

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/insecure
# version : flight 1995
#
############################################################
```

NAME

insecure - enable the write on a few partitions

SYNOPSIS

insecure

DESCRIPTION

insecure enable the write ation on the partitions containing the following data:

```
-partition address: 018000H     => SCRIPT SEGMENT

-                   018000H     => HV_SEG(mobc)
                                       // CALIB_SEG(cfmp)

-                   01C000H     => RFDC_SEG(mobc)

-                   01D000H     => CONST
```

EXAMPLES

The insecure command must be executed before a calibration:

```
insecure
xxx -m xx xx
xxx -M xx xx
secure
```

The insecure command must be executed before loading/modifying a script

```
insecure
rm scriptname
secure
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/kill
# version : flight 1995
#
############################################################
```

NAME

kill - kills a command or script that runs in background

SYNOPSIS

```
kill task_id
kill -i
```

DESCRIPTION

kill task_id	kills the command or script that runs in the subshell with id : 'task_id'.
kill -i	lists the name, id, priority and status of all created tasks.

'kill task_id' stops any command or script that has been started in background (& after the command or scriptname). To each of this background commands or scripts a subshell is assigned with each a different 'task_id'.

The kill command will be acknowledged by the following line :

"Background task to be killed : 'task_id'"

When the command or script has been killed the following line will appear :

"Killed in task 'task_id'"

The user has to specify the task_id of the subshell in which the command or script, that he wants to kill, runs.
Therefore 'kill -i' provides a list of all existing tasks.

For each task a 'name', an 'id', a 'priority' and a 'status' is given.
'kill -i' functions similarly to the 'ps'-command in UNIX.

'status' can take the following values :

0	:	explicitly suspended	[sc_tsuspend()]

2	:	suspended for message	[sc_pend()]
4	:	suspended for input	[sc_getc()]
8	:	suspended for output	[sc_putc()]
16	:	awaiting special character	[sc_waitc()]
32	:	suspended for task delay	[sc_delay()]
64	:	suspended on message queue	[sc_qpend()]

EXAMPLES

kill 12	kills command or script running in subshell with task_id = 12
kill -i	gives the following info list :

```
Shell ..... Id : 1 , Prior : 10 , Stat : 0
Watchdog .. Id : 2 , Prior : 9 , Stat : 16
Stripper .. Id : 3 , Prior : 12 , Stat : 96
Subshell .. Id : 10 , Prior : 11 , Stat : 64
Subshell .. Id : 11 , Prior : 11 , Stat : 64
Subshell .. Id : 12 , Prior : 11 , Stat : 64
```

REMARKS

'kill' can not kill commands or scripts that run in foreground. More than that, you will never get a prompt during execution of a foreground command or script. This implies that you won't be able to execute any command what so ever, until the running command or script has stopped.

To kill foreground commands or scripts, use ctrl_C (see corresponding man page).

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/let
# version : flight 1995
#
############################################################
```

NAME

let - changes the value of a global variable

SYNOPSIS

let NAME = VALUE1±VALUE2

DESCRIPTION

The let command can only be used for global variables with integer values.

The command assigns a new value to the global variable NAME.

If NAME does not exist yet, it is created.

This new value is calculated from the expression "VALUE1±VALUE2" VALUEj and VALUEk can be integer numbers, but can also be the values of global variables (others than NAME or NAME itself).

When VALUE1 and/or VALUE2 are global variables they are referenced in the classic way, i.e. their name preceeded by the dollar sign.

EXAMPLES

Imagine three global variables exist :

```
A=12
B=34
C=105
```

The following combinations are possible :

```
let A = 100+23          ----> A=123
let A = $A+100          ----> A=112
let A = 100-$A          ----> A=88
let B = $B+$B           ----> B=68
let B = $C-$A           ----> B=93

etc...
let
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/rm
# version : flight 1995
#
############################################################
```

NAME

rm - remove a script from the script directory

SYNOPSIS

rm scriptname

DESCRIPTION

rm removes the entry for a script from the script directory.

The scriptname may contain upto 15 printable characters.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/scripts
# version : flight 1995
#
############################################################
```

SUBJECT

scripts

DESCRIPTION

A script is a named list of commands.
A script can be executed by using its name as a command.
The list of commands in the script will be executed line by line.

A script can be created with the command
cat > SCRIPTNAME
This command opens a line editor enabling the user to enter the commands line by line. To leave the editor (i.e. to end the script) a dot has to be introduced on the last scriptline.

A script can be read with the command :
cat SCRIPTNAME

A list of existing scripts (directory) is given by the "dir"-command.

A script can be copied to another by using :
cp SCRIPTNAME1 SCRIPTNAME2

A script can be removed from the directory :
rm SCRIPTNAME

The complete directory can be cleaned up with the "fsck"-command.

All these commands have their own manual page.

Scripts can be called from within scripts (nesting of scripts). It is advised not to nest scripts more than four levels deep.
Be careful! Loops, appearing in some scripts are also looked upon as nesting levels.

For the use of loops (for- and while-loops) in scripts refer to the manual page "loops".

It is possible to pass parameters to scripts (positional parameters).
This gives the user the possibility to write scripts in a very general way. The combination of global variables and positional parameters opens a broad range of combinations feasible with minimal commands and/or scripts.

A scripts parameterlist can contain maximum 9 parameters.
The syntax is as follows :
SCRIPTNAME [p1 [p2 ... [p9]]]]]]]]]
In the script these parameters are referenced by their number of appearance in the list, preceeded by the dollar sign :
p1 is referenced as $1
p2 is referenced as $2
...
p9 is referenced as $9

p1, p2, ..., p9 are called the positional parameters.
These positional parameters can be of all types (numbers, strings, global variables). Of course, using a positional parameter which references a string in a command which expects an integer, will yield an error.

It is advised not to use $1,...,$9 from the command line or to use 1,...,9 as names for global variables.

EXAMPLES

Positional parameters :

Imagine the global variables :
TIME=testscript
WAIT=1000

Imagine a script (named "test") containing the following

```
echo $4
sleep $1
date $5 $2 34 $3
```

The script could be called as follows :

```
test $WAIT 12 0 $TIME 127
```

First "testscript" would be printed, then a 10 second sleep would be executed and finaly the date would be set to "127 12 34 00".

REMARKS

It is advised not to run long duration scripts in foreground in CFMP, in order not to block MOBC (will await acknoledge character that is sent only after script termination).
However, it is possible to start a long duration foreground scripts in CFMP if it is followed immediately by Ctrl_C

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/secure
# version : flight 1995
#
############################################################
```

NAME

secure - protect a few partitions of RAM

SYNOPSIS

secure

DESCRIPTION

secure disable the write ation on the partitions containing the following data:

-partition address: 018000H => SCRIPT SEGMENT

\- 018000H => HV_SEG(mobc)
// CALIB_SEG(cfmp)

\- 01C000H => RFDC_SEG(mobc)

\- 01D000H => CONST

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/set
# version : flight 1995
#
############################################################
```

NAME

set - lists the existing global variables

SYNOPSIS

set

DESCRIPTION

This command lists the existing global variables in the form :

```
NAME1   :       VALUE1
NAME2   :       VALUE2
```

At the end of the list the number of existing global variables is shown, as well as the number of global variables that still can be created. A maximum of 10 global variables can be created.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/sleep
# version : flight 1995
#
############################################################
```

NAME

sleep - suspend execution for a time interval

SYNOPSIS

sleep time

DESCRIPTION

'sleep' suspends execution for time/100 seconds.
Time is given in ticks of a 100 Hz clock.

It can be useful when put in a script after a command that changes a voltage.
'sleep' then permits the voltage to become stable before execution continues.

EXAMPLES

sleep 3000 suspends execution for 30 sec.

```
#############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/unset
# version : flight 1995
#
#############################################################
```

NAME

unset - deletes one or all global variables

SYNOPSIS

```
unset
unset NAME
```

DESCRIPTION

This command offers the possibility to purge one global variable from the global variables list, or to clean up the entire list.

EXAMPLES

`unset`	clears the complete list of global variables
`unset EXAMPLE`	purges the global variable "EXAMPLE" from the list

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/both/whileloop
# version : flight 1995
#
############################################################
```

NAME

while-loops in scripts

SYNOPSIS

```
while test VARLEFT comp VARRIGHT
do
command 1
command 2
...
command n
done
```

with comp in {-eq,-ne,-le,-ge,-lt,-gt}

DESCRIPTION

When a script has been opened for editing (see manual page 'cat') it is possible to introduce a while-loop, enabling repeated executions of commands using a comparison procedure between to parameters VARLEFT and VARRIGHT.
The loop consists of an 'opening' line and the loop body.
The first line of the loop body is always "do", the last line "done".

The opening line contains two variables VARLEFT and VARRIGHT.
At least one of these two variables has to be a global variable (existing or not existing) of the form $GV. The other value can be also a global variable or an integer number.
Possibilities : while test $GV comp VALUE
while test VALUE comp $GV
while test $GVi comp $GVj
All the global variables used in the opening line (minimum one and maximum two) will not be deleted after the last iteration.

The opening line contains furthermore a comparator "comp".
This comparator can take one of 6 expressions : -eq, -ne, -le, -ge, -lt or -gt.
The loop will be iterated until the condition in the opening line "VARLEFT comp VARRIGHT" is met.

For most applications it will be necessary to include at the end of the body of the loop, a "let"-command that changes the value of one of the variables in order to meet after a while the specified condition.

It is possible to create endless loops by specifying conditions that will never be met (e.g. by omitting the "let"-command at the end of the loop body) or by setting one of the variables to an extremely high value.

Nesting of loops is possible (while-loops in while-loops and while- and for-loops) but not indefinitely. It is advised not to nest more than four levels.
Be careful! Also calling a script from a script is one level of nesting.

EXAMPLES

Imagine a script "example" containing the following lines :

```
A=0
while test $A -le 500
do
echo $A
let A = $A+100
done
```

Executing the script will output :

```
0
100
200
300
400
500
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/chk
# version : flight 1995
#
############################################################
```

NAME

chk - measures the chemical housekeeping values and
prints them to the telecommand (TC) echo,
or
sends them to the telemeasurement (TM) channel.

SYNOPSIS

```
chk     : prints the chemical housekeeping summary
          to the TC echo.
chk -tm : sends the chemical housekeeping values
          to the TM channel.
```

DESCRIPTION

chk : printing of a summary of the chemical housekeeping values, measured by reading the housekeeping ADC. For each quantity two values are printed, i.e. an hexadecimal value (16 bits) and a decimal value (12 least significant bits of the 16 bit ADC value.

DAC value = mxxx nnnn nnnn nnnn

where m = 0 indicates a valid ADC measurement,
m = 1 indicates an invalid ADC measurement,
x = don't care,
n = one of the 12 bits that contain the real measured housekeeping value.

The summary looks as follows :

CHEMICAL HOUSEKEEPING SUMMARY

	Hex (16 bit)	Dec (12 bit)
R: Thermistor at Cl flush flow :	aaaa	bbbb
R: Thermistor at Cl flow :	aaaa	bbbb
R: Thermistor at Flush valve :	aaaa	bbbb
R: Thermistor at Cl valve :	aaaa	bbbb
R: Thermistor at I flush flow :	aaaa	bbbb
R: Thermistor at I flow :	aaaa	bbbb

R: Thermistor at I valve :	aaaa	bbbb
R: Thermistor at flange :	aaaa	bbbb
R: Thermistor of air amb. up :	aaaa	bbbb
R: Thermistor of air a. down :	aaaa	bbbb
R: Thermistor at motor body :	aaaa	bbbb
R: Thermistor at mot. control :	aaaa	bbbb
R: Internal Pressure :	aaaa	bbbb
R: Air Compressed Pressure :	aaaa	bbbb
R: Lower R reference :	aaaa	bbbb
R: Upper R reference :	aaaa	bbbb
R Status of cl:		
R ----------		
R: Valve in status :	aaaa	bbbb
R: Valve out status :	aaaa	bbbb
R: Flow :	aaaa	bbbb
R: Low pressure :	aaaa	bbbb
R: High pressure :	aaaa	bbbb
R Status of I :		
R ----------		
R: Valve in status :	aaaa	bbbb
R: Valve out status :	aaaa	bbbb
R: Flow :	aaaa	bbbb
R: Low pressure :	aaaa	bbbb
R: High pressure :	aaaa	bbbb
R Flush Status :		
R ----------		
R: valve :	aaaa	bbbb
R: Cl flow :	aaaa	bbbb
R: I Flow :	aaaa	bbbb
R: Low pressure :	aaaa	bbbb
R: High pressure :	aaaa	bbbb
R Discharge Status :		
R ----------		
R: Discharge Current :	aaaa	bbbb
R: Discharge voltage :	aaaa	bbbb
R Lamp Status :		
R ----------		
R: Lamp Current :	aaaa	bbbb
R: Lamp voltage :	aaaa	bbbb
R Control of turbo pump:		
R ----------		
R: Speed :	aaaa	bbbb
R: Current :	aaaa	bbbb
R: Temperature :	aaaa	bbbb
R: Battery voltage :	aaaa	bbbb

where aaaa = a 16 bit value in hex form (0000 --> FFFF).
(if aaaa > 7FFF (i.e. MSBit =1) then invalid).
bbbb = aaaa & 0x0FFF = masking of the 4 MSBits of aaaa.

chk -tm : sends the 16 bit DAC values to the TM channel.

EXAMPLES

```
chk
chk -tm
```

```
##############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/dis
# version : flight 1995
#
##############################################################
```

NAME

dis - set or control the discharge high voltage in the flow

SYNOPSIS

```
dis
dis off
dis cont
dis pulse
dis -t
dis -px
dis -pa
dis value
dis +  step
dis -r dacvalue
dis -m mindacvalue minvalue
dis -M maxdacvalue maxvalue
```

DESCRIPTION

dis	return measured value of current and voltage
dis off	switch discharge off
dis cont	switch discharge on
dis pulse	set discharge in pulse mode for tof measurement
dis -t	list the settings
dis -px	print only hex and dec value (after "dis")
dis -pa	print also real float value (after "dis")
dis value	set discharge voltage to a real value
dis + step	increment discharge voltage by step
dis -r dacvalue	set dis dac to integer dacvalue
dis -m mindacvalue minvalue	set minimum dac and discharge value
dis -M maxdacvalue maxvalue	set maximum dac and discharge value

It is assumed that:


```
dacvalue        = a signed integer number between 0 and 255
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

value           = a floating point number in C's %f or %e format
minvalue        = the minimum dis allowed for normal operation
maxvalue        = the maximum dis allowed for normal operation
```

The dis command with the -r, -m, or -M option is normally only used during testing and initial configuration of the discharge supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the dis supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
  0 yields      0 Volt output
255 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the dis configuration is saved.

The dis command without an option and with a floating point number as an argument should be used to set dis to a value without knowing the discharge board characteristics.

The dis command with a + option and with a floating point number as an argument should be used to increment dis by a positive or negative value without knowing the discharge board characteristics.

EXAMPLES

dis cont	to switch on the dis supply
dis -r 200	to measure dis with a 200 dac value
dis -m 10 300.0	to set the relation dac to dis
dis -M 250 2000.0	to set the relation dac to dis
dis -t	to see the configuration
dis -px	preceeding "dis"-command
dis -pa	preceeding "dis"-command
dis 1752.9	to set dis to 1752.9 Volt
dis + 250.0	to increment dis by 250 Volt
dis + -11.8	to decrement dis by 11.8 Volt
dis	to measure discharge voltage and current
dis off	to switch off the discharge supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/flow
# version : flight 1995
#
############################################################
```

NAME

flow - set or control the output of the flow meters.

SYNOPSIS

```
flow X on
flow X off

flow | Cl
     | I    | off
     | XCl  | on
     | XI   | -t
            -px
            -pa
            flow
            +  flowstep
            -r dacvalue
            -m mindacvalue minflowvalue
            -M maxdacvalue maxflowvalue
```

DESCRIPTION

```
flow X on                 switch the two flush flow on
flow X off                switch the two flush flow off


flow ...                  return measured value of specified
                             flow.
flow ... off              switch specified flow off
flow ... on               switch specified flow on
flow ... -t               list the settings
flow ... -px              print only hex
                             and dec flow value (after "flow")
flow ... -pa              print also real
                             float flow value (after "flow")
flow ... flowvalue        set flow to a real flowvalue
flow ... +  flowstep      increment flow by flowstep
flow ... -r dacvalue      set flow dac to integer dacvalue
flow ... -m mindacvalue minflowvalue
                          set minimum dac and flow value
flow ... -M maxdacvalue maxflowvalue
                          set maximum dac and flow value
```


It is assumed that:

```
dacvalue        = a signed integer number between 0 and 255
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

flowvalue       = a floating point number in C's %f or %e format
minflowvalue    = the minimum flowvalue allowed for normal operation
maxflowvalue    = the maximum flowvalue allowed for normal operation
```

The flow command with the -r, -m, or -M option is normally only used during testing and initial configuration of the flow supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the flow supply in a direct way and to check the relationship between different dac settings and the flow voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
 0 yields       0 Volt output
 255 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and flow voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the flow configuration is saved.

The flow command without an option and with a floating point number as an argument should be used to set flow to a value without knowing the flow board characteristics.

The flow command with a + option and with a floating point number as an argument should be used to increment flow by a positive or negative value without knowing the flow board characteristics.

EXAMPLES

```
flow X on                      to switch on the flow
flow I -r 200            to measure flow with a 200 dac value
flow Cl -m 1 0           to set the relation dac to flow
flow XCl -M  255 1000    to set the relation dac to flow
flow XCl -t              to see the configuration
flow I -px               preceeding "flow"-command
flow Cl -pa              preceeding "flow"-command
flow I 190               to set flow to 190 sccm
flow I + 200             to increment flow by 200 sccm
flow I + -180            to decrement floflow by 180 sccm
flow Cl                  to measure flow
flow XCl off             to switch off the flow
```

REMARKS

The physical output voltage line for flow has a value in the range
0 Volt to +10 Volt, corresponding to a flow between 0 and 1000 sccm.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/lamp
# version : flight 1995
#
############################################################
```

NAME

lamp - set or control the lamp high voltage in the flow

SYNOPSIS

```
lamp
lamp off
lamp cont
lamp pulse
lamp -t
lamp -px
lamp -pa
lamp value
lamp +  step
lamp -r dacvalue
lamp -m mindacvalue minvalue
lamp -M maxdacvalue maxvalue
```

DESCRIPTION

lamp	return measured value of current and voltage
lamp off	switch lamp off
lamp cont	switch lamp on
lamp pulse	set lamp in lamp mode for tof measurement
lamp -t	list the settings
lamp -px	print only hex and dec value (after "lamp")
lamp -pa	print also real float value (after "lamp")
lamp value	set lamp voltage to a real value
lamp + step	increment lamp voltage by step
lamp -r dacvalue	set lamp dac to integer dacvalue
lamp -m mindacvalue minvalue	set minimum dac and lamp value
lamp -M maxdacvalue maxvalue	set maximum dac and lamp value

It is assumed that:

dacvalue = a signed integer number between 0 and 255

mindacvalue = the minimum dacvalue allowed for normal operation
maxdacvalue = the maximum dacvalue allowed for normal operation

value = a floating point number in C's %f or %e format
minvalue = the minimum lamp voltage allowed for normal operation
maxvalue = the maximum lamp voltage allowed for normal operation

The lamp command with the -r, -m, or -M option is normally only used during testing and initial configuration of the lamp supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the lamp supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
  0 yields      0 Volt output
255 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the lamp configuration is saved.

The lamp command without an option and with a floating point number as an argument should be used to set lamp to a value without knowing the lamp board characteristics.

The lamp command with a + option and with a floating point number as an argument should be used to increment lamp by a positive or negative value without knowing the lamp board characteristics.

EXAMPLES

lamp cont	to switch on the lamp supply
lamp -r 200	to measure lamp with a 200 dac value
lamp -m 10 300.0	to set the relation dac to lamp
lamp -M 250 2000.0	to set the relation dac to lamp
lamp -t	to see the configuration
lamp -px	preceeding "lamp"-command
lamp -pa	preceeding "lamp"-command
lamp 1752.9	to set lamp to 1752.9 Volt
lamp + 250.0	to increment lamp voltage by 250 Volt
lamp + -11.8	to decrement lamp voltage by 11.8 Volt
lamp	to measure lamp voltage and current
lamp off	to switch off the lamp supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/pres
# version : flight 1995
#
############################################################
```

NAME

pres - enables, disables and measures pressure sensor:
- 'pap' This sensor measures the pressurised air pressure (range 0 - 10 bar)
- 'flush gas'
- 'Cl'
- 'I'

SYNOPSIS

pres	Cl	
	I	on
	X	off

DESCRIPTION

pres ... : measures the ADC value of the pressure sensor
pres ... on : enables the power supply of the pressure sensor.
pres ... off : disables the power supply of the pressure sensor.

When ... is X, it include the flush sensor and the 'pap' sensor.

EXAMPLES

```
pres X
pres I on
pres Cl off
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/turbof
# version : flight 1995
#
############################################################
```

NAME

turbof - control the turbopump and set the frequency

SYNOPSIS

```
turbof
turbof off
turbof on
turbof -t
turbof -px
turbof -pa
turbof turbofvalue
turbof +  turbofstep
turbof -r dacvalue
turbof -m mindacvalue minturbofvalue
turbof -M maxdacvalue maxturbofvalue
```

DESCRIPTION

```
turbof                    return measured turbof value
turbof off                switch turbof off
turbof on                 switch turbof on
turbof -t                 list the settings
turbof -px                print only hex and dec turbof
                             value (after "turbof")
turbof -pa                print also real float turbof
                             value (after "turbof")
turbof turnofvalue        set turbof to a real turbofvalue
turbof +  turbofstep      increment turbof by turbofstep
turbof -r dacvalue        set turbof dac to integer dacvalue
turbof -m mindacvalue minturbofvalue
                          set minimum dac and turbof value
turbof -M maxdacvalue maxalue
                          set maximum dac and turbof value
```

It is assumed that:

```
dacvalue        = a signed integer number between 0 and 4096
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

turbofvalue     = a floating point number in C's %f or %e format
```



```
minturbofvalue  = the minimum turbofvalue allowed for normal operation
maxturbofvalue  = the maximum turbofvalue allowed for normal operation
```

The turbof command with the -r, -m, or -M option is normally only used during testing and initial configuration of the turbof supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the turbof supply in a direct way and to check the relationship between different dac settings and voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
   0 yields      0 Volt output (0 Hz)
4096 yields      10 Volt output (1500 Hz)
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the turbof configuration is saved.

The turbof command without an option and with a floating point number as an argument should be used to set turbof to a value without knowing the turbof board characteristics.

The turbof command with a + option and with a floating point number as an argument should be used to increment turbof by a positive or negative value without knowing the turbof board characteristics.

EXAMPLES

```
turbof on               to switch on the turbof supply
turbof -r 200           to measure turbof with a 200 dac value
turbof -m 10 300.0      to set the relation dac to turbof
turbof -M 4000 2000.0   to set the relation dac to turbof
turbof -t               to see the configuration
turbof -px              preceeding "turbof"-command
turbof -pa              preceeding "turbof"-command
turbof 1752.9           to set turbof to 1752.9 Volt
turbof + 250.0          to increment turbof by 250 Volt
turbof + -11.8          to decrement turbof by 11.8 Volt
turbof                  to measure turbof
turbof off              to switch off the turbof supply
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/turboi
# version : flight 1995
#
############################################################
```

NAME

turboi - control the turbopump and set the maximal current

SYNOPSIS

```
turboi
turboi off
turboi on
turboi -t
turboi -px
turboi -pa
turboi turboivalue
turboi +  turboistep
turboi -r dacvalue
turboi -m mindacvalue minturboivalue
turboi -M maxdacvalue maxturboivalue
```

DESCRIPTION

turboi	return measured turboi value
turboi off	switch turboi off
turboi on	switch turboi on
turboi -t	list the settings
turboi -px	print only hex and dec turboi value (after "turboi")
turboi -pa	print also real float turboi value (after "turboi")
turboi turnofvalue	set turboi to a real turboivalue
turboi + turboistep	increment turboi by turboistep
turboi -r dacvalue	set turboi dac to integer dacvalue
turboi -m mindacvalue minturboivalue	set minimum dac and turboi value
turboi -M maxdacvalue maxalue	set maximum dac and turboi value

It is assumed that:

```
dacvalue        = a signed integer number between 0 and 4096
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

turboivalue     = a floating point number in C's %f or %e format
```



```
minturboivalue  = the minimum turboivalue allowed for normal operation
maxturboivalue  = the maximum turboivalue allowed for normal operation
```

The turboi command with the -r, -m, or -M option is normally only used during testing and initial configuration of the turboi supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the turboi supply in a direct way and to check the relationship between different dac settings and voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
   0 yields      0 Amp output
4096 yields      5 Amp output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the turboi configuration is saved.

The turboi command without an option and with a floating point number as an argument should be used to set turboi to a value without knowing the turboi board characteristics.

The turboi command with a + option and with a floating point number as an argument should be used to increment turboi by a positive or negative value without knowing the turboi board characteristics.

EXAMPLES

turboi on	to switch on the turboi supply
turboi -r 200	to measure turboi with a 200 dac value
turboi -m 10 300.0	to set the relation dac to turboi
turboi -M 4000 2000.0	to set the relation dac to turboi
turboi -t	to see the configuration
turboi -px	preceeding "turboi"-command
turboi -pa	preceeding "turboi"-command
turboi 1752.9	to set turboi to 1752.9 Volt
turboi + 250.0	to increment turboi by 250 Volt
turboi + -11.8	to decrement turboi by 11.8 Volt
turboi	to measure turboi
turboi off	to switch off the turboi supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/cfmp/valve
# version : flight 1995
#
############################################################
```

NAME

cl - opens, closes, asks status and measures value of
Cl, I, Flush Cl and Flush I pneumatic valves.

SYNOPSIS

valve	Cl	in
	I	out
		close
		status
valve	X	close
		open
		status

DESCRIPTION

valve Cl in : opens the Cl in valve and close the Cl out valve.
valve Cl out : opens the Cl out valve and close the Cl in valve.
valve I close: close the I in and I out valves.
valve I status: asks the status of the two I valves.
valve X close: close the Flush valve.
valve X open: open the Flush valve.
valve X status: asks the status of the flush valve.

The possible values returned by 'status' are :

```
< 2800              : " Valve closed" (0 Volt);
>=2800 and < 3400   : " Valve opened" (24 Volt);
>=3400              : " Valve defect" (28 Volt);
```

EXAMPLES

```
valve I status
valve X open
valve Cl out
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/ahk
# version : flight 1995
#
############################################################
```

NAME

ahk - measures the ambient housekeeping values and
prints them to the telecommand (TC) echo,
or
sends them to the telemeasurement (TM) channel.

SYNOPSIS

ahk : prints the ambient housekeeping summary
to the TC echo.
ahk -tm : sends the ambient housekeeping values
to the TM channel.

DESCRIPTION

ahk : printing of a summary of the ambient housekeeping values, measured by reading the housekeeping ADC. For each quantity two values are printed, i.e. an hexadecimal value (16 bits) and a decimal value (12 least significant bits of the 16 bit ADC value.

DAC value = mxxx nnnn nnnn nnnn

where m = 0 indicates a valid ADC measurement,
m = 1 indicates an invalid ADC measurement,
x = don't care,
n = one of the 12 bits that contain the real measured housekeeping value.

The summary looks as follows :

AMBIENT HOUSEKEEPING SUMMARY

		Hex (16 bit)	Dec (12 bit)
Thermistor at north side	:	aaaa	bbbbb
Thermistor at east side	:	aaaa	bbbbb
Thermistor at south side	:	aaaa	bbbbb
Thermistor at west side	:	aaaa	bbbbb
Pressure by mks10	:	aaaa	bbbbb

Pressure by mks100	:	aaaa	bbbbb
Lower R reference	:	aaaa	bbbbb
Upper R reference	:	aaaa	bbbbb

where aaaa = a 16 bit value in hex form (0000 --> FFFF).
(if aaaa > 7FFF (i.e. MSBit =1) then invalid).
bbbb = aaaa & 0x0FFF = masking of the 4 MSBits of aaaa.

ahk -tm : sends the 16 bit DAC values to the TM channel.

EXAMPLES

```
ahk
ahk -tm
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/cap
# version : flight 1995
#
############################################################
```

NAME

cap - instruction set to command the opening of the two shutters.
(the inlet shutter and the outlet shutter of the flowtube)

SYNOPSIS

cap	Asks the status of the shutters (opto couplers)
cap test	Testing the wiring of the circuit.
cap enable	Arming of the circuit : ARMMOS is put ON. A software flag ENABLE is set.
cap confirm	Confirmation of the arming. An additional software flag CONFIRM is set.
cap open	Firing of the shutters : two FIREMOS are put ON. One for each shutter, with a time interval of two seconds. The inlet shutter opens first, two seconds later the outlet shutter opens.
cap disable	Sets MOSFETs and flags to their initial values, i.e. ARMMOS = OFF FIREMOSIN = OFF FIREMOSOUT = OFF ENABLE = 0 CONFIRM = 0

The opening procedure of the shutters is carefully secured.
Any combination of cap commands other than the three commands
"cap enable, cap confirm and cap open", directly following each
other, will cause the circuit to be put in the initial
configuration
(i.e. ARMMOS OFF, FIREMOSIN OFF and FIREMOSOUT OFF,
ENABLE and CONFIRM flags unset).
E.g. : "cap enable, cap confirm, cap confirm", etc. will
reinitialize the circuit.

DESCRIPTION

```
cap         : asks status of the shutters.
cap test    : tests circuit wiring.
cap disable : sets MOSFETs and flags to their initial values.
cap enable  : arms the circuit.
cap confirm : software confirmation of the arming.
```



```
cap open    : fires the shutters : first the inlet shutter, two seconds
              later the outlet shutter.
```

EXAMPLES

```
cap
cap test
cap disable
cap enable
cap confirm
cap open
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/check
# version : flight 1995
#
############################################################
```

NAME

check - Calculate checksum of specified segment

SYNOPSIS

check

DESCRIPTION

check	Display checksum of important segments

EXAMPLES

check

ANSWER is :

```
checksum for scripts segment    : 4589
checksum for rfdc table segment : -24312
checksum for hv table segment   : -23824
checksum for constants segment  : 12699
checksum for code segment       : -14627
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/cll
# version : flight 1995
#
############################################################
```

NAME

cll - put octopole power supply in closed loop

SYNOPSIS

cll

DESCRIPTION

This commands puts the octopole power supply into closed loop.

This is done by giving the fifth switch (number 4) on the

The command "cll" is equivalent to the C-function call "rfdc((int)4,"on")"

Remark : the first four switches on the rfdc card are used to switch the four rfdc supplies :

- switch 0 : iiv
- switch 1 : opb
- switch 2 : rf
- switch 3 : aux

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/cpv
# version : flight 1995
#
############################################################
```

NAME

cpv - set or control the channel plate high voltage supply

SYNOPSIS

```
cpv
cpv off
cpv on
cpv -t
cpv -px
cpv -pa
cpv hvvalue
cpv +  hvstep
cpv -r dacvalue
cpv -m mindacvalue minhvvalue
cpv -M maxdacvalue maxhvvalue
```

DESCRIPTION

`cpv`	return measured cpv value
`cpv off`	switch cpv off
`cpv on`	switch cpv on
`cpv -t`	list the settings
`cpv -px`	print only hex and dec hv value (after "cpv")
`cpv -pa`	print also real float hv value (after "cpv")
`cpv hvvalue`	set cpv to a real hvvalue
`cpv +  hvstep`	increment cpv by hvstep
`cpv -r dacvalue`	set cpv dac to integer dacvalue
`cpv -m mindacvalue minhvvalue`	set minimum dac and hv value
`cpv -M maxdacvalue maxhvvalue`	set maximum dac and hv value

It is assumed that:

```
dacvalue        = a signed integer number between 0 and -8192
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

hvvalue         = a floating point number in C's %f or %e format
minhvvalue      = the minimum hvvalue allowed for normal operation
maxhvvalue      = the maximum hvvalue allowed for normal operation
```


The cpv command with the -r, -m, or -M option is normally only used during testing and initial configuration of the cpv supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the cpv supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
    0 yields      0 Volt output
-8192 yields    -10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the cpv configuration is saved.

The cpv command without an option and with a floating point number as an argument should be used to set cpv to a value without knowing the cpv board characteristics.

The cpv command with a + option and with a floating point number as an argument should be used to increment cpv by a positive or negative value without knowing the cpv board characteristics.

EXAMPLES

`cpv on`	to switch on the cpv supply
`cpv -r -2000`	to measure cpv with a -2000 dac value
`cpv -r -700`	to measure cpv with a -700 dac value
`cpv -m -700 300.0`	to set the relation dac to cpv
`cpv -M -5000 2000.0`	to set the relation dac to cpv
`cpv -t`	to see the configuration
`cpv -px`	preceeding "cpv"-command
`cpv -pa`	preceeding "cpv"-command
`cpv 1752.9`	to set cpv to 1752.9 Volt
`cpv + 250.0`	to increment cpv by 250 Volt
`cpv + -11.8`	to decrement cpv by 11.8 Volt
`cpv`	to measure cpv
`cpv off`	to switch off the cpv supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/gauge
# version : flight 1995
#
############################################################
```

NAME

gauge - enables and disables vacuum gauge

SYNOPSIS

```
gauge on
gauge off
```

DESCRIPTION

```
gauge on   : enables  the power supply of the vacuum gauge
gauge off  : disables the power supply of the vacuum gauge
```

EXAMPLES

```
gauge on
gauge off
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/grid
# version : flight 1995
#
############################################################
```

NAME

grid - set or control the output of the octopole RF supply

SYNOPSIS

```
grid
grid off
grid on
grid -t
grid -px
grid -pa
grid gridvalue
grid +  gridstep
grid -r dacvalue
grid -m mindacvalue mingridvalue
grid -M maxdacvalue maxgridvalue
```

DESCRIPTION

grid	return measured grid value
grid off	switch grid off
grid on	switch grif on
grid -t	list the settings
grid -px	print only hex and dec rfdc value (after "grid")
grid -pa	print also real float rfdc value (after "grid")
grid gridvalue	set grid to a real gridvalue
grid + gridstep	increment grid by gridstep
grid -r dacvalue	set grid dac to integer dacva
grid -m mindacvalue mingridvalue	set minimum dac and grid valu
grid -M maxdacvalue maxgridvalue	set maximum dac and grid valu

It is assumed that:

```
dacvalue        = a signed integer number between -8192 and 8192
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

gridvalue       = a floating point number in C's %f or %e format
mingridvalue    = the minimum gridvalue allowed for normal operation
maxgridvalue    = the maximum gridvalue allowed for normal operation
```


The grid command with the -r, -m, or -M option is normally only used during testing and initial configuration of the grid supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the grid supply in a direct way and to check the relationship between different dac settings and the grid voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
-8192 yields    -10 Volt output
 8192 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and grid voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the grid configuration is saved.

The grid command without an option and with a floating point number as an argument should be used to set grid to a value without knowing the grid board characteristics.

The grid command with a + option and with a floating point number as an argument should be used to increment grid by a positive or negative value without knowing the grid board characteristics.

EXAMPLES

grid on	to switch on the grid
grid -r -2000	to measure grid with a -2000 dac value
grid -r -700	to measure grid with a -700 dac value
grid -m -8192 -10.0	to set the relation dac to grid
grid -M 8192 10.0	to set the relation dac to grid
grid -t	to see the configuration
grid -px	preceeding "grid"-command
grid -pa	preceeding "grid"-command
grid 1.9	to set grid to 1.9 Volt
grid + 2.0	to increment grid by 2 Volt
grid + -1.8	to decrement grid by 1.8 Volt
grid	to measure grid
grid off	to switch off the grid

REMARKS

The physical output voltage line for grid has a value in the range -10 Volt to +10 Volt.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/hvgen
# version : flight 1995
#
############################################################
```

NAME

hvgen - general switch for high voltages

SYNOPSIS

```
hvgen off
hvgen on
```

DESCRIPTION

hvgen off	switches all high voltages off
hvgen on	hvgen has to be switched on in order to be able to switch on the high voltages separately (i.e. itv, psv, pav, cpv)

EXAMPLES

hvgen on	general high voltage switch on
hvgen off	general high voltage switch off

```
#############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/iiv
# version : flight 1995
#
#############################################################
```

NAME

iiv - set or control the ion inlet voltage

SYNOPSIS

```
iiv
iiv off
iiv on
iiv -t
iiv -px
iiv -pa
iiv iivvalue
iiv +  iivstep
iiv -r dacvalue
iiv -m mindacvalue miniivvalue
iiv -M maxdacvalue maxiivvalue
```

DESCRIPTION

iiv	return measured iiv value
iiv off	switch iiv off
iiv on	switch iiv on
iiv -t	list the settings
iiv -px	print only hex and dec rfdc value (after "iiv")
iiv -pa	print also real float rfdc value (after "iiv")
iiv iivvalue	set iiv to a real iivvalue
iiv + iivstep	increment iiv by iivstep
iiv -r dacvalue	set iiv dac to integer dacvalue
iiv -m mindacvalue miniivvalue	set minimum dac and iiv value
iiv -M maxdacvalue maxiivvalue	set maximum dac and iiv value

It is assumed that:

```
dacvalue       = a signed integer number between -8192 and 8192
mindacvalue    = the minimum dacvalue allowed for normal operation
maxdacvalue    = the maximum dacvalue allowed for normal operation

iivvalue       = a floating point number in C's %f or %e format
miniivvalue    = the minimum iivvalue allowed for normal operation
maxiivvalue    = the maximum iivvalue allowed for normal operation
```


The iiv command with the -r, -m, or -M option is normally only used during testing and initial configuration of the iiv supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the iiv supply in a direct way and to check the relationship between different dac settings and the iiv voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
-8192 yields    -10 Volt output
 8192 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and iiv voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the iiv configuration is saved.

The iiv command without an option and with a floating point number as an argument should be used to set iiv to a value without knowing the iiv board characteristics.

The iiv command with a + option and with a floating point number as an argument should be used to increment iiv by a positive or negative value without knowing the iiv board characteristics.

EXAMPLES

iiv on	to switch on the iiv
iiv -r -2000	to measure iiv with a -2000 dac value
iiv -r -700	to measure iiv with a -700 dac value
iiv -m -8192 -10.0	to set the relation dac to iiv
iiv -M 8192 10.0	to set the relation dac to iiv
iiv -t	to see the configuration
iiv -px	preceeding "iiv"-command
iiv -pa	preceeding "iiv"-command
iiv 1.9	to set iiv to 1.9 Volt
iiv + 2.0	to increment iiv by 2 Volt
iiv + -1.8	to decrement iiv by 1.8 Volt
iiv	to measure iiv
iiv off	to switch off the iiv

REMARKS

The physical output voltage line for iiv has a value in the range -10 Volt to +10 Volt.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/itv
# version : flight 1995
#
############################################################
```

NAME

itv - set or control the inner toroidal high voltage supply

SYNOPSIS

```
itv
itv off
itv on
itv -t
itv -px
itv -pa
itv hvvalue
itv +  hvstep
itv -r dacvalue
itv -m mindacvalue minhvvalue
itv -M maxdacvalue maxhvvalue
```

DESCRIPTION

itv	return measured itv value
itv off	switch itv off
itv on	switch itv on
itv -t	list the settings
itv -px	print only hex and dec hv value (after "itv")
itv -pa	print also real float hv value (after "itv")
itv hvvalue	set itv to a real hvvalue
itv + hvstep	increment itv by hvstep
itv -r dacvalue	set itv dac to integer dacvalue
itv -m mindacvalue minhvvalue	set minimum dac and hv value
itv -M maxdacvalue maxhvvalue	set maximum dac and hv value

It is assumed that:

```
dacvalue      = a signed integer number between 0 and -8192
mindacvalue   = the minimum dacvalue allowed for normal operation
maxdacvalue   = the maximum dacvalue allowed for normal operation

hvvalue       = a floating point number in C's %f or %e format
minhvvalue    = the minimum hvvalue allowed for normal operation
maxhvvalue    = the maximum hvvalue allowed for normal operation
```


The itv command with the -r, -m, or -M option is normally only used during testing and initial configuration of the itv supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the itv supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
    0 yields      0 Volt output
-8192 yields    -10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the itv configuration is saved.

The itv command without an option and with a floating point number as an argument should be used to set itv to a value without knowing the itv board characteristics.

The itv command with a + option and with a floating point number as an argument should be used to increment itv by a positive or negative value without knowing the itv board characteristics.

EXAMPLES

itv on	to switch on the itv supply
itv -r -2000	to measure itv with a -2000 dac value
itv -r -700	to measure itv with a -700 dac value
itv -m -700 300.0	to set the relation dac to itv
itv -M -5000 2000.0	to set the relation dac to itv
itv -t	to see the configuration
itv -px	preceeding "itv"-command
itv -pa	preceeding "itv"-command
itv 1752.9	to set itv to 1752.9 Volt
itv + 250.0	to increment itv by 250 Volt
itv + -11.8	to decrement itv by 11.8 Volt
itv	to measure itv
itv off	to switch off the itv supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/mks100
# version : flight 1995
#
############################################################
```

NAME

mks100 - enables and disables pressure sensor 'mks100'.

SYNOPSIS

```
mks100 on
mks100 off
```

DESCRIPTION

mks100 on : enables the power supply of the 'msks100' pressure sensor.
mks100 off : disables the power supply of the 'msks100' pressure sensor.

EXAMPLES

```
mks100 on
mks100 off
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/msp
# version : flight 1995
#
############################################################
```

NAME

msp - get mass spectrum from or control T4 processor

SYNOPSIS

```
msp -l discrilevel
msp -s
msp  numberofspectra
msp -t numberofspectra
msp -b numberofspectra
```

DESCRIPTION

msp -l discrilevel	set discriminator level
msp -s	get T4 and T6 status
msp number	get spectrum with "number" accumulations
msp -t number	get testspectrum" with "number" accumulations
msp -b number	get backgroundspectrum with "number" accumulations

It is assumed that:

number = an unsigned integer number between 0 and 65535
discrilevel = an unsigned integer number between 0 and 65535

When one of the first two commands is issued, the T4-processor returns a status frame (type 4). When one of the three last commands is given, the T4 processor returns a frame (type 7), containing a digital as well as an analog spectrum. The MOBC separates digital from analog data, and transmits them separately to the TM-channel. The analog part is given type number 2, the digital part type number 12 (see soft for flight May 1991).
The analog spectrum and the digital spectrum are transmitted to TM in blocks of 256 bytes. Therefore a "stripper" program exists, cutting the spectrum in blocks and defining these blocks by creating an adapted header.
When the command "msp number" is issued the T4-mode is : 400
When the command "msp -t number" is issued the T4-mode is : 98
When the command "msp -b number" is issued the T4-mode is : 99

For further details on the working principals of the T4 processor read the T4 manual (Urs Jenzer).

EXAMPLES

msp -l 50	set discriminatorlevel to 50
msp -s	get status of T4 and T6
msp 1000	get spectrum with 1000 accumulations
msp -t 2000	get sumspectrum with 2000 accumulations
msp -b 689	get backgroundspectrum with 689 accumulations

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/opb
# version : flight 1995
#
############################################################
```

NAME

opb - set or control the octopole bias voltage

SYNOPSIS

```
opb
opb off
opb on
opb -t
opb -px
opb -pa
opb opbvalue
opb +  opbstep
opb -r dacvalue
opb -m mindacvalue minopbvalue
opb -M maxdacvalue maxopbvalue
```

DESCRIPTION

```
opb                                 return measured opbvalue
opb off                             switch opb off
opb on                              switch opb on
opb -t                              list the settings
opb -px                             prints only hex and dec
                                    rfdc value (after "opb")
opb -pa                             prints also real float
                                    rfdc value (after "opb")
opb opbvalue                        set opb to a real opbvalue
opb +  opbstep                      increment opb by opbstep
opb -r dacvalue                     set opb dac to integer dacvalue
opb -m mindacvalue minopbvalue      set minimum dac and opb value
                              rem: itv must be itv=0
opb -M maxdacvalue maxopbvalue      set maximum dac and opb value
                              rem: itv must be itv=0
```

It is assumed that:

```
dacvalue        = a signed integer number between -8192 and 8192
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

opbvalue        = a floating point number in C's %f or %e format
```



```
minopbvalue     = the minimum opbvalue allowed for normal operation
maxopbvalue     = the maximum opbvalue allowed for normal operation
```

The opb command with the -r, -m, or -M option is normally only used during testing and initial configuration of the opb supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the opb supply in a direct way and to check the relationship between different dac settings and the opb voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
-8192 yields    -10 Volt output
 8192 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and opb voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the opb configuration is saved.

The opb command without an option and with a floating point number as an argument should be used to set opb to a value without knowing the opb board characteristics.

The opb command with a + option and with a floating point number as an argument should be used to increment opb by a positive or negative value without knowing the opb board characteristics.

EXAMPLES

```
opb on                  to switch on the opb
opb -r -2000            to measure opb with a -2000 dac value
opb -r -700             to measure opb with a  -700 dac value
opb -m -8192 -10.0      to set the relation dac to opb
opb -M  8192 10.0       to set the relation dac to opb
opb -t                  to see the configuration
opb -px                 preceeding "opb"-command
opb -pa                 preceeding "opb"-command
opb 1.9                 to set opb to 1.9 Volt
opb + 2.0               to increment opb by 2 Volt
opb + -1.8              to decrement opb by 1.8 Volt
opb                     to measure opb
opb off                 to switch off the opb
```

REMARKS

The output voltage at the DAC for the octopole bias is opb=POLEBIAS*2/3. The physical output voltage line for opb is opb*3/2+iiv (due to hardware) or POLEBIAS+iiv (min -25 Volt, max 25 Volt).

```
#############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/pav
# version : flight 1995
#
#############################################################
```

NAME

pav - set or control the post acceleration high voltage supply

SYNOPSIS

```
pav
pav off
pav on
pav -t
pav -px
pav -pa
pav hvvalue
pav +  hvstep
pav -r dacvalue
pav -m mindacvalue minhvvalue
pav -M maxdacvalue maxhvvalue
```

DESCRIPTION

pav	return measured pav value
pav off	switch pav off
pav on	switch pav on
pav -t	list the settings
pav -px	print only hex and dec hv value (after "pav")
pav -pa	print also real float hv value (after "pav")
pav hvvalue	set pav to a real hvvalue
pav + hvstep	increment pav by hvstep
pav -r dacvalue	set pav dac to integer dacvalue
pav -m mindacvalue minhvvalue	set minimum dac and hv value
pav -M maxdacvalue maxhvvalue	set maximum dac and hv value

It is assumed that:

```
dacvalue        = a signed integer number between 0 and -8192
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

hvvalue         = a floating point number in C's %f or %e format
minhvvalue      = the minimum hvvalue allowed for normal operation
maxhvvalue      = the maximum hvvalue allowed for normal operation
```


The pav command with the -r, -m, or -M option is normally only used during testing and initial configuration of the pav supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the pav supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
    0 yields      0 Volt output
-8192 yields    -10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the pav configuration is saved.

The pav command without an option and with a floating point number as an argument should be used to set pav to a value without knowing the pav board characteristics.

The pav command with a + option and with a floating point number as an argument should be used to increment pav by a positive or negative value without knowing the pav board characteristics.

EXAMPLES

pav on	to switch on the pav supply
pav -r -2000	to measure pav with a -2000 dac value
pav -r -700	to measure pav with a -700 dac value
pav -m -700 300.0	to set the relation dac to pav
pav -M -5000 2000.0	to set the relation dac to pav
pav -t	to see the configuration
pav -px	preceeding "pac"-command
pav -pa	preceeding "pac"-command
pav 1752.9	to set pav to 1752.9 Volt
pav + 250.0	to increment pav by 250 Volt
pav + -11.8	to decrement pav by 11.8 Volt
pav	to measure pav
pav off	to switch off the pav supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/psv
# version : flight 1995
#
############################################################
```

NAME

psv - set or control the phosphor screen high voltage supply

SYNOPSIS

```
psv
psv off
psv on
psv -t
psv -px
psv -pa
psv hvvalue
psv +  hvstep
psv -r dacvalue
psv -m mindacvalue minhvvalue
psv -M maxdacvalue maxhvvalue
```

DESCRIPTION

psv	return measured psv value
psv off	switch psv off
psv on	switch psv on
psv -t	list the settings
psv -px	print only hex and dec hv value (after "psv")
psv -pa	print also real float hv value (after "psv")
psv hvvalue	set psv to a real hvvalue
psv + hvstep	increment psv by hvstep
psv -r dacvalue	set psv dac to integer dacvalue
psv -m mindacvalue minhvvalue	set minimum dac and hv value
psv -M maxdacvalue maxhvvalue	set maximum dac and hv value

It is assumed that:

```
dacvalue      = a signed integer number between 0 and -8192
mindacvalue   = the minimum dacvalue allowed for normal operation
maxdacvalue   = the maximum dacvalue allowed for normal operation

hvvalue       = a floating point number in C's %f or %e format
minhvvalue    = the minimum hvvalue allowed for normal operation
maxhvvalue    = the maximum hvvalue allowed for normal operation
```


The psv command with the -r, -m, or -M option is normally only used during testing and initial configuration of the psv supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the psv supply in a direct way and to check the relationship between different dac settings and high voltage values as measured by a high voltage voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
    0 yields      0 Volt output
-8192 yields    -10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and high voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the psv configuration is saved.

The psv command without an option and with a floating point number as an argument should be used to set psv to a value without knowing the psv board characteristics.

The psv command with a + option and with a floating point number as an argument should be used to increment psv by a positive or negative value without knowing the psv board characteristics.

EXAMPLES

psv on	to switch on the psv supply
psv -r -2000	to measure psv with a -2000 dac value
psv -r -700	to measure psv with a -700 dac value
psv -m -700 300.0	to set the relation dac to psv
psv -M -5000 2000.0	to set the relation dac to psv
psv -t	to see the configuration
psv -px	preceeding "psv"-command
psv -pa	preceeding "psv"-command
psv 1752.9	to set psv to 1752.9 Volt
psv + 250.0	to increment psv by 250 Volt
psv + -11.8	to decrement psv by 11.8 Volt
psv	to measure psv
psv off	to switch off the psv supply

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/rf
# version : flight 1995
#
############################################################
```

NAME

rf - set or control the output of the octopole RF supply

SYNOPSIS

```
rf
rf off
rf on
rf -t
rf -px
rf -pa
rf rfvalue
rf +  rfstep
rf -r dacvalue
rf -m mindacvalue minrfvalue
rf -M maxdacvalue maxrfvalue
```

DESCRIPTION

rf	return measured rf value
rf off	switch rf off
rf on	switch rf on
rf -t	list the settings
rf -px	print only hex and dec rfdc value (after "rf")
rf -pa	print also real float rfdc value (after "rf")
rf rfvalue	set rf to a real rfvalue
rf + rfstep	increment rf by rfstep
rf -r dacvalue	set rf dac to integer dacvalue
rf -m mindacvalue minrfvalue	set minimum dac and rf value
rf -M maxdacvalue maxrfvalue	set maximum dac and rf value

It is assumed that:

```
dacvalue        = a signed integer number between -8192 and 8192
mindacvalue     = the minimum dacvalue allowed for normal operation
maxdacvalue     = the maximum dacvalue allowed for normal operation

rfvalue = a floating point number in C's %f or %e format
minrfvalue      = the minimum rfvalue allowed for normal operation
maxrfvalue      = the maximum rfvalue allowed for normal operation
```


The rf command with the -r, -m, or -M option is normally only used during testing and initial configuration of the rf supply.

The -r (raw) option allows to control the digital-to-analog converter associated with the rf supply in a direct way and to check the relationship between different dac settings and the rf voltage values as measured by a voltmeter.

Nominal relationship between dacvalue and dac output voltage:

```
-8192 yields    -10 Volt output
 8192 yields     10 Volt output
```

Subsequently the -m (minimum) and -M (Maximum) options are used to define the linear relationship between dac settings and rf voltages based on the previous measurements .

The -t (table) option lists the contents of the table wherein the rf configuration is saved.

The rf command without an option and with a floating point number as an argument should be used to set rf to a value without knowing the rf board characteristics.

The rf command with a + option and with a floating point number as an argument should be used to increment rf by a positive or negative value without knowing the rf board characteristics.

EXAMPLES

rf on	to switch on the rf
rf -r -2000	to measure rf with a -2000 dac value
rf -r -700	to measure rf with a -700 dac value
rf -m -8192 -10.0	to set the relation dac to rf
rf -M 8192 10.0	to set the relation dac to rf
rf -t	to see the configuration
rf -px	preceeding "rf"-command
rf -pa	preceeding "rf"-command
rf 1.9	to set rf to 1.9 Volt
rf + 2.0	to increment rf by 2 Volt
rf + -1.8	to decrement rf by 1.8 Volt
rf	to measure rf
rf off	to switch off the rf

REMARKS

The physical output voltage line for rf has a value in the range -10 Volt to +10 Volt.

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/thk
# version : flight 1995
#
############################################################
```

NAME

thk - measures the technological housekeeping values and
prints them to the telecommand (TC) echo,
or
sends them to the telemeasurement (TM) channel.

SYNOPSIS

thk : prints the technological housekeeping summary
to the TC echo.
thk -tm : sends the technological housekeeping values
to the TM channel.

DESCRIPTION

thk : printing of a summary of the technological housekeeping values, measured by reading the housekeeping ADC. For each quantity two values are printed, i.e. an hexadecimal value (16 bits) and a decimal value (12 least significant bits of the 16 bit ADC value.

DAC value = mxxx nnnn nnnn nnnn

where m = 0 indicates a valid ADC measurement,
m = 1 indicates an invalid ADC measurement,
x = don't care,
n = one of the 12 bits that contain the real measured housekeeping value.

The summary looks as follows :

TECHNOLOGICAL HOUSEKEEPING SUMMARY

		Hex (16 bit)	Dec (12 bit)
Thermistor at battery	:	aaaa	bbbbb
Thermistor at flange	:	aaaa	bbbbb
Thermistor at octopole supply	:	aaaa	bbbbb
Thermistor at electronics	:	aaaa	bbbbb
Internal pressure	:	aaaa	bbbbb

Ion pump	:	aaaa	bbbbb
Lower R reference	:	aaaa	bbbbb
Upper R reference	:	aaaa	bbbbb
Wiring test	:	aaaa	bbbbb
Cap tot. closed detector	:	aaaa	bbbbb
Cap tot. open detector	:	aaaa	bbbbb

---> c = conclusion for cap status

Valve position detector	:	aaaa	bbbbb

---> v = conclusion for valve status

where aaaa = a 16 bit value in hex form (0000 --> FFFF).
(if aaaa > 7FFF (i.e. MSBit =1) then invalid).
bbbb = aaaa & 0x0FFF = masking of the 4 MSBits of aaaa.
c = cap closed,
cap open,
cap moving or not in a final position,
cap ordetectors malfunctioning,
impossible detector values.
v = valve moving,
valve closed,
valve open,
valve malfunctioning.

thk -tm : sends the 16 bit DAC values to the TM channel.

EXAMPLES

thk
thk -tm

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/tof
# version : flight 1995
#
############################################################
```

NAME

tof - get time of flight spectrum

SYNOPSIS

```
tof -s
tof  numberofspectra
tof -d delay
tof -r rate
tof -s status
tof -g gain1 gain2
```

DESCRIPTION

`tof -s`	get delay, rate, period status, gain
`tof  number`	get spectrum with "number of accumulations"
`tof -d number`	set the delay
`tof -r number`	set the rate
`tof -p number`	set the period
`tof -g number1 number2`	set the 2 gain:

It is assumed that:

number = an unsigned integer number between 0 and 65535

When the second command is issued, the MOBC generate a tof spectrum that is composed of a report packet (type 7), then 16 spectrum packet (type 8) containing 64 pixel of the tof spectrum. Those packets are transmitted to TM in blocks of 256 bytes.

number1, number2 = 1,2,4,8

gain = number1 * number2

EXAMPLES

`tof -s`	ask status of PIT and actual gain
`tof  500`	take a tof spectrum with 500 accumulations
`tof -d 15000`	program rhe delay to 15 msec
`tof -r 100`	program the rate to 100 µsec
`tof -p 1000`	program the period to 1 msec
`tof -g 8 1`	program the gain of the instrumentation amplifiers to 8 rep 1

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/commands/mobc/valve
# version : flight 1995
#
############################################################
```

NAME

valve - opens, closes or asks status of valve between the high and the low vacuum.

SYNOPSIS

```
valve open
valve close
valve status
```

DESCRIPTION

valve open : opens the valve.
valve close : closes the valve.
valve status : asks the status of the valve.
The possible values returned by 'valve status' are :

```
    0 : "Valve moving";
    8 : "Valve closed";
   32 : "Valve opened";
other : "Valve malfunctioning";
```

EXAMPLES

```
valve open
valve close
valve status
```

```
#############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/errors/errorlist
# version : flight 1995
#
#############################################################
```

```
*****************************************************************************
*  Errors appear on the screen under the next form :                        *
*                                                                           *
*       'E xxy'                                                             *
*                                                                           *
*  where xx are two characters indicating the command in which an error     *
*  was detected, and where y is a sequence number.                          *
*  xxy is called the ERROR CODE.                                            *
*                                                                           *
*  Warnings appear on the screen under the next form :                      *
*                                                                           *
*       'W xxy'                                                             *
*                                                                           *
*  where xx are two characters indicating the command for which a           *
*  warning was made, and where y is a sequence number.                      *
*  xxy is called the WARNING CODE.                                          *
*                                                                           *
*****************************************************************************
```

I. ERRORS

ERROR CODE	ERROR MESSAGE	SOURCEFILE
ca1	cat command : impossible to cat two files.	parser.c
ca3	cat command : impossible to cat 0 files.	parser.c
ca4	cat command : script name unknown.	cat.c
ca5	cat command : no global variables allowed.	parser.c
cap	cap command : wrong combination of cap commands	cap.c
ch1	check command : invalid number of arguments	check.c
co1	long contains unrecognizable character.	ltoa.c
cf1	chk -tm command: pend on mailbox	cfmp.c
cf2	chk -tm command: crc error in data from cfmp	cfmp.c
cl1	flcl or hvcl command : slope not properly set.	discharge.c
cl2	flcl or hvcl command : usage wrong.	discharge.c
cl3	flcl or hvcl command : adcmin = adcmax ==> slope = x/0.	discharge.c

cl4	cl command : usage wrong	discharge.c
cl5	clhp or cllp command : invalid number of arguments	discharge.c
co1	conversion error from long to string	ltoa.c
cp1	cp : aborted. Source script name unknown.	cp.c
cp2	cp : aborted. Destination script already exists.	cp.c
cp3	cp : aborted. Script table full.	cp.c
cp4	cp : aborted. Scriptfile full.	cp.c
cp5	cp command has too much arguments.	parser.c
ct1	flctc or hvctc command : slope not properly set.	lamp.c
ct2	flctc or hvctc command : usage wrong.	lamp.c
ct3	flctc or hvctc command : adcmin = adcmax ==> slope = x/0.	lamp.c
ct4	ctc command : usage wrong	lamp.c
ct5	pap or ctcp command : invalid number of arguments	lamp.c
da2	date command : usage wrong. Use 'date' to get the time and 'date DDD HH MM SS' to set the time.	date.c
di1	dis command : usage of parameters wrong	disch.c
di2	dis command : adcmin = adcmax ==> slope = x/0	disch.c
ec1	echo commands do not contain any argument	echo.c
ex1	no "do"-line in "for"-loop.	execbody.c
ex2	no "in"-keyword in "for"-command.	execbody.c
ex3	variable in "for"-command could not be set properly.	execbody.c
ex4	"list" in "for"-command contains 0 or more then 10 arguments.	execbody.c
ex5	wrong "for"-command syntax.	execbody.c
ex6	no "do"-line in "while"-loop.	execbody.c
ex7	no "test"-keyword in "while"-command.	execbody.c
ex8	global variable in left member of "while"-loop not found.	execbody.c
ex9	global variable in right member of "while"-loop not found.	execbody.c
ex10	comparator in "while"-loop wrong.	
ex11	"while"-command syntax wrong.	execbody.c
ex12	the command line contains too much positional parameters.	execbody.c
ex13	no scripts exist.	execscript.c
ex14	scriptname not recognized.	execscript.c
ex15	global variable in "list" in "for"-command. not found	execbody.c
fl1	flow command : usage of parameters wrong	flow.c
fl1	flow command : adcmin = adcmax ==> slope = x/0	flow.c
fl3	flow command : mismatch in adc min and max	compval.c
ga1	gauge command : wrong usage.	gauge.c

ga2	mks100 command : wrong usage.	mks100.c
gv1	global variable not found	printgv.c delgv.c
gv2	global variable table full	setgv.c letgv.c
hk1	ahk or thk command : invalid housekeeping number.	hsk.c(mobc)
hk2	ahk or thk command : too much, too long or invalid parameters.	hsk.c(mobc)
hk3	chk command : too much, too long or invalid parameters.	hsk.c(cfmp)
hv1	hv command : slope not properly set.	hv.c
hv2	hv command : usage wrong.	hv.c
hv3	hv command : adcmin = adcmax ==> slope = x/0.	hv.c
ki1	kill command : usage wrong.	kill.c
la1	lamp command : usage of parameters wrong	lamp.c
la2	lamp command : adcmin = adcmax ==> slope = x/0	lamp.c
ma1	cll command : can not be executed from main	main.c
ms1	msp command : frame queue time out.	msp.c
ms2	msp command : f11 CRC error.	msp.c
ms3	msp command : compression error.	msp.c
ms4	msp command : f1 CRC error.	msp.c
ms5	msp command : usage wrong.	msp.c
ms6	msp command : compression error.	msp.c
pa4	unknown global variable in parameter list.	parser.c echo.c letgv.c
pa5	wrong command or invalid number of parameters.	parser.c
pa6	wrong number of positional parameters.	parser.c
pa7	command "cll" not successful	main.c
pa9	echo command contains more then one argument and first argument is gloval variable or positional parameter	echo.c
pr1	pres command : invalid parameterlist	pressure.c
rm1	rm command : no argument specified	rm.c
rm2	rm command : too many/few or too long argument(s).	parser.c
rm4	rm aborted. Script name unknown.	rm.c
rm5	rm aborted. no scripts exist.	rm.c
rf1	rf/dc command : slope not properly set.	rfdc.c
rf2	rf/dc command : usage wrong.	rfdc.c
rf3	rfdc command : adcmin = adcmax ==> slope = x/0.	rfdc.c
sh1	execution impossible : command contains unexpected characters.	shell.c

sh2	asynchronous execution impossible : line partition exhausted.	shell.c
sh3	asynchronous execution impossible : subshell queue full.	shell.c
sh4	line not successfully released.	subshell.c
sl1	sleep command : usage wrong.	sleep.c
st1	queue error or timeout on queue to stripper	stripper.c
tf1	tof command : unrecognizable parameter	newtof.c
to1	to : aborted. Script exists already.	to.c
to2	to command : script table or scriptfile already full.	to.c
to3	to : aborted. No script stored.	to.c
to4	to command : cannot be executed. Script overflow.	to.c
tu1	turbo command : slope not properly set.	turbo.c
tu2	turbo command : usage wrong.	turbo.c
tu3	turbo command : adcmin=adcmax ==> slope = x/0.	turbo.c
va1	valve command : invalid parameterlist	valve.c (cfmp)
va2	valve command : invalid parameterlist	valve.c (cfmp)
vl1	valve command : usage wrong.	valve.c

II. WARNINGS

****

WARNING CODE	WARNING MESSAGE	SOURCEFILE
da1	date command : clock has not been set yet. Use 'date DDD HH MM SS' to set clock.	date.C
W atof	float format is not correct	atof.c
W atoi	integer format is not correct	atoi.c
W atol	long format is not correct	atol.c

Telemetingpakketten

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid12
# version : flight 1995
#
############################################################
```

DIGITAL SPECTRUM PACKET

```
---------------------- general structure -----------------------------
Packet header   : 8 bytes

byte 0          : packet ID (0..255)
byte 1          : packet specific flags
bytes 2 and 3   : packet specific 16 bit packet sequence number
bytes 4 to 7    : 32 bit time tag (units are 10 msec)

Data field      : packet specific data

CRC field       : 16 bit cyclic redundancy check over all preceeding bytes


---------------------- packet specific -------------------------------

packet ID               = 12 (indicates digital spectrum packet)

data field[0..1]        = 16 bit spectrum number as generated by the MOBC
data field[2..3]        = spectrum block number (0 to 7)
data field[4..259]      = 1 block of 256 bytes of spectrum data

               # 1 spectrum           = 1024 pixel values of 16 bits
               #                      = 2048 bytes
               #                      = 8 spectrum blocks of 256 bytes

total packet length     = 8 + 4 + 256 + 2 = 270 bytes

packet duration         = 270 bytes / 960 bytes.sec-1 = 281 msec @ 9600 baud
                        = 270 bytes / 480 bytes.sec-1 = 563 msec @ 4800 baud

               # spectrum transmission time    = 2.25 sec @ 9600 baud
               #                               = 4.50 sec @ 4800 baud

Remark : this packet (type 12) is always send to the groundstation in
combination with a analog spectrum packet (type 2).

               # transmission time for a "double" spectrum (mixed)

               # At 9600 baud
               # ------------
```



```
               # (16+8) packets @  281 msec  = 6.75 sec

               # At 4800 baud
               # ------------
               # (16+8) packets @  563 msec  = 13.50 sec
~
~
~


               Total for "double" spectrum (mixed)
               ***********************************

               At 9600 baud
               ------------
               (16+8) packets @  281 msec  = 6.75 sec

               At 4800 baud
               ------------
               (16+8) packets @  563 msec  = 13.50 sec
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid2
# version : flight 1995
#
############################################################
```

ANALOG SPECTRUM PACKET

```
---------------------- general structure -------------------------------
Packet header   : 8 bytes

byte 0          : packet ID (0..255)
byte 1          : packet specific flags
bytes 2 and 3   : packet specific 16 bit packet sequence number
bytes 4 to 7    : 32 bit time tag (units are 10 msec)

Data field      : packet specific data

CRC field       : 16 bit cyclic redundancy check over all preceeding bytes


---------------------- packet specific --------------------------------

packet ID               = 2 (indicates analog spectrum packet)

data field[0..1]        = 16 bit spectrum number as generated by the MOBC
data field[2..3]        = spectrum block number (0 to 15)
data field[4..259]      = 1 block of 256 bytes of spectrum data


               # 1 spectrum    = 1024 pixel values of 32 bits
               #               = 4096 bytes
               #               = 16 spectrum blocks of 256 bytes

total packet length     = 8 + 4 + 256 + 2 = 270 bytes

packet duration         = 270 bytes / 960 bytes.sec-1 = 281 msec @ 9600 baud
                        = 270 bytes / 480 bytes.sec-1 = 563 msec @ 4800 baud

               # spectrum transmission time    = 4.50 sec @ 9600 baud
               #                               = 9.00 sec @ 4800 baud


Remark : this packet (type 2) is always send to the groundstation in
combination with a digital spectrum packet (type 12).

               # transmission time for a "double" spectrum (mixed)
```



```
               # At 9600 baud
               # ------------
               # (16+8) packets @  281 msec  = 6.75 sec

               # At 4800 baud
               # ------------
               # (16+8) packets @  563 msec  = 13.50 sec
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid33
# version : flight 1995
#
############################################################
```

SPECTRUM REPORT PACKET

---------- general structure ----------

Packet header : 8 bytes

byte 0 : packet ID (0..255)
byte 1 : packet specific flags
bytes 2 and 3 : packet specific 16 bit packet sequence number
bytes 4 to 7 : 32 bit time tag (units are 10 msec)

Data field : packet specific data

CRC field : 16 bit cyclic redundancy check over all preceeding bytes

---------- packet specific ----------

packet ID = 33 (indicates spectrum report packet)

data field[0..1] = 16 bit = spectrum number as generated by the MOBC
data field[2..3] = 16 bit = number of accumulations in spec as provided by T4
data field[4..7] = 32 bit = inner toroidal voltage measurement
data field[8..11] = 32 bit = post acceleration voltage measurement
data field[12..15] = 32 bit = channel plate voltage measurement
data field[16..19] = 32 bit = phosphor screen voltage measurement
data field[20..23] = 32 bit = ion inlet voltage measurement
data field[24..27] = 32 bit = octopole pole bias measurement
data field[28..31] = 32 bit = octopole rf measurement
data field[32..35] = 32 bit = auxilliary measurement
data field[34..37] = 16 bit = measurement mode as provided by T4
data field[38..39] = 16 bit = inner toroidal voltage setting
data field[40..41] = 16 bit = post acceleration voltage setting
data field[42..43] = 16 bit = channel plate voltage setting
data field[44..45] = 16 bit = phosphor screen voltage setting
data field[46..47] = 16 bit = ion inlet voltage setting
data field[48..49] = 16 bit = octopole pole bias setting
data field[50..51] = 16 bit = octopole rf setting
data field[52..53] = 16 bit = auxilliary setting

total packet length = 8 + 54 + 2 = 64 bytes

packet duration = 64 bytes / 960 bytes.sec-1 = 66.6 msec @ 9600 baud

= 64 bytes / 480 bytes.sec-1 = 133.3 msec @ 4800 baud

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid4
# version : flight 1995
#
############################################################
```

PIXELPROCESSOR STATUS PACKET

```
------------------------- general structure -----------------------------
Packet header   : 8 bytes

byte 0          : packet ID (0..255)
byte 1          : packet specific flags
bytes 2 and 3   : packet specific 16 bit packet sequence number
bytes 4 to 7    : 32 bit time tag (units are 10 msec)

Data field      : packet specific data

CRC field       : 16 bit cyclic redundancy check over all preceeding bytes

------------------------- packet specific -------------------------------

packet ID               = 4 (indicates T4/T6 status packet)

data field[0]           =  8 bits leading zero
data field[1]           =  8 bits actual spectrum type
data field[2]           =  8 bits program version
data field[3..6]        = 32 bits spectrum number
data field[7..10]       = 32 bits flight time
data field[11..12]      = 16 bits mode of operation
data field[13..14]      = 16 bits discriminator level
data field[15..16]      = 16 bits error 1
     ...                      ...
     ...                      ...
data field[63..64]      = 16 bits error 25
data field[65..66]      = 16 bits portout 1
data field[67..68]      = 16 bits portout 2
data field[69..70]      = 16 bits portout 3
data field[71..72]      = 16 bits portout 4
data field[73..74]      = 16 bits portin 1
data field[75..76]      = 16 bits portin 2
data field[77..78]      = 16 bits portin 3
data field[79..80]      = 16 bits portin 4
data field[81..82]      = 16 bits houskepping adc 1
     ...                      ...
     ...                      ...
data field[111..112]    = 16 bits houskepping adc 16
```



```
total packet length     = 8 + 113 + 2 = 123 bytes

packet duration         = 123 bytes / 960 bytes.sec-1 = 128.1 msec @ 9600 baud
                        = 123 bytes / 480 bytes.sec-1 = 256.3 msec @ 4800 baud
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid5
# version : flight 1995
#
############################################################
```

TECHNOLOGICAL HOUSEKEEPING PACKET

---------- general structure ----------

Packet header : 8 bytes

byte 0 : packet ID (0..255)
byte 1 : packet specific flags
bytes 2 and 3 : packet specific 16 bit packet sequence number
bytes 4 to 7 : 32 bit time tag (units are 10 msec)

Data field : packet specific data

CRC field : 16 bit cyclic redundancy check over all preceeding bytes

---------- packet specific ----------

packet ID = 5 (indicates technological housekeeping packet)

data field[0..1] = right justified 12 bit val of internal thermistor at battery [int_therm_bat](Fenwal)
data field[2..3] = right justified 12 bit val of internal thermistor at flange [int_therm_flange](Fenwal)
data field[4..5] = right justified 12 bit val of internal thermistor at octopole supply [int_therm_octsup](Fenwal)
data field[6..7] = right justified 12 bit val of internal thermistor at electronics [int_therm_elec](Fenwal)
data field[8..9] = right justified 12 bit val of internal thermistor at detector A [int_therm_det_A](Fenwal)
data field[10..11] = right justified 12 bit val of internal thermistor at detector B [int_therm_det_B](Fenwal)
data field[12..13] = right justified 12 bit val of inlet cap
data field[14..15] = right justified 12 bit val of outlet cap
data field[16..17] = right justified 12 bit val of inlet wiring test
data field[18..19] = right justified 12 bit val of outlet wiring test
data field[20..21] = right justified 12 bit val of internal pressure
data field[22..23] = right justified 12 bit val of lower R reference
data field[24..25] = right justified 12 bit val of upper R reference
data field[26..27] = right justified 12 bit val of ion pump
data field[28..29] = val (0,8,32) of valve position

total packet length = 8 + 30 + 2 = 40 bytes

packet duration = 40 bytes / 960 bytes.sec-1
= 41.6 msec @ 9600 baud

= 40 bytes / 480 bytes.sec-1
= 83.3 msec @ 4800 baud

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid6
# version : flight 1995
#
############################################################
```

AMBIENT HOUSEKEEPING PAKCKET

---------- general structure ----------

Packet header : 8 bytes

byte 0 : packet ID (0..255)
byte 1 : packet specific flags
bytes 2 and 3 : packet specific 16 bit packet sequence number
bytes 4 to 7 : 32 bit time tag (units are 10 msec)

Data field : packet specific data

CRC field : 16 bit cyclic redundancy check over all preceeding bytes

---------- packet specific ----------

packet ID = 6 (indicates ambient housekeeping packet)

data field[0..1] = right justified 12 bit val of ambient thermistor at north side (VIZ)
data field[2..3] = right justified 12 bit val of ambient thermistor at east side (Fenwal)
data field[4..5] = right justified 12 bit val of ambient thermistor at south side (VIZ)
data field[6..7] = right justified 12 bit val of ambient thermistor at west side (Fenwal)
data field[8..9] = right justified 12 bit val of ambient pressure with 100 mbar range (mks100)
data field[10..11] = right justified 12 bit val of lower R reference
data field[12..13] = right justified 12 bit val of upper R reference

total packet length = 8 + 14 + 2 = 24 bytes

packet duration = 24 bytes / 960 bytes.sec-1
= 25.0 msec @ 9600 baud

= 24 bytes / 480 bytes.sec-1
= 50.0 msec @ 4800 baud

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid7
# version : flight 1995
#
############################################################
```

TIME OF FLIGHT REPORT PACKET

---------- general structure ----------

Packet header : 8 bytes

byte 0 : packet ID (0..255)
byte 1 : packet specific flags
bytes 2 and 3 : packet specific 16 bit packet sequence number
bytes 4 to 7 : 32 bit time tag (units are 10 msec)

Data field : packet specific data

CRC field : 16 bit cyclic redundancy check over all preceeding bytes

---------- packet specific ----------

packet ID = 7 (indicates tof report packet)

data field[0..1] = 16 bit = spectrum number as generated by the MOBC
data field[2..3] = 16 bit = number of accumulations in spec as provided by T4
data field[4..7] = 32 bit = inner toroidal voltage measurement
data field[8..11] = 32 bit = post acceleration voltage measurement
data field[12..15] = 32 bit = channel plate voltage measurement
data field[16..19] = 32 bit = phosphor screen voltage measurement
data field[20..23] = 32 bit = ion inlet voltage measurement
data field[24..27] = 32 bit = octopole pole bias measurement
data field[28..31] = 32 bit = octopole rf measurement
data field[32..35] = 32 bit = auxilliary measurement
data field[36..37] = 16 bit = tof parameter delay
data field[38..39] = 16 bit = tof parameter rate
data field[40..41] = 16 bit = tof parameter period
data field[42..43] = 16 bit = tof parameter gain
data field[44..45] = 16 bit = inner toroidal voltage setting
data field[46..47] = 16 bit = post acceleration voltage setting
data field[48..49] = 16 bit = channel plate voltage setting
data field[50..51] = 16 bit = phosphor screen voltage setting
data field[52..53] = 16 bit = ion inlet voltage setting
data field[54..55] = 16 bit = octopole pole bias setting
data field[56..57] = 16 bit = octopole rf setting
data field[58..59] = 16 bit = auxilliary setting

total packet length = 8 + 60 + 2 = 70 bytes

packet duration = 70 bytes / 960 bytes.sec-1 = 72.9 msec @ 9600 baud
= 70 bytes / 480 bytes.sec-1 = 145.8 msec @ 4800 baud

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid8
# version : flight 1995
#
############################################################
```

TIME OF FLIGHT SPECTRUM PACKET

---------- general structure ----------

Packet header : 8 bytes

byte 0 : packet ID (0..255)
byte 1 : packet specific flags
bytes 2 and 3 : packet specific 16 bit packet sequence number
bytes 4 to 7 : 32 bit time tag (units are 10 msec)

Data field : packet specific data

CRC field : 16 bit cyclic redundancy check over all preceeding bytes

---------- packet specific ----------

packet ID = 8 (indicates tof spectrum packet)

data field[0..1] = 16 bit spectrum number as generated by the MOBC
data field[2..3] = spectrum block number (0 to 7)
data field[4..259] = 1 block of 256 bytes of spectrum data

```
# 1 spectrum          = 1024 pixel values of 16 bits
#                     = 2048 bytes
#                     = 8 spectrum blocks of 256 bytes
```

total packet length = 8 + 4 + 256 + 2 = 270 bytes

packet duration = 270 bytes / 960 bytes.sec-1 = 281 msec @ 9600 baud
= 270 bytes / 480 bytes.sec-1 = 563 msec @ 4800 baud

```
# spectrum transmission time    = 2.25 sec @ 9600 baud
#                               = 4.50 sec @ 4800 baud
~
~
~
```

Total for "double" spectrum (mixed)
****

At 9600 baud

(16+8) packets @ 281 msec = 6.75 sec

At 4800 baud

(16+8) packets @ 563 msec = 13.50 sec

```
#################################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/packetid9
# version : flight 1995
#
#################################################################
```

CHEMICAL HOUSEKEEPING PACKET

```
------------------------- general structure -----------------------------
Packet header   : 8 bytes

byte 0          : packet ID (0..255)
byte 1          : packet specific flags
bytes 2 and 3   : packet specific 16 bit packet sequence number
bytes 4 to 7    : 32 bit time tag (units are 10 msec)

Data field      : packet specific data

CRC field       : 16 bit cyclic redundancy check over all preceeding bytes

------------------------- packet specific -------------------------------

packet ID               = 9 (indicates chemical housekeeping packet)

data field[0..3]        = 32 bit time tag (units are 10 msec)
                          of the cfmp.
data field[4..5]        = packet specific 16 bit packet sequence number
                          of the cfmp.
data field[6..7]        = right justified 12 bit val of thermistor at flush
                          flowmeter in Cl-branch
data field[8..9]        = right justified 12 bit val of thermistor at Cl
                          flowmeter
data field[10..11]      = right justified 12 bit val of thermistor at flush
                          valve
data field[12..13]      = right justified 12 bit val of thermistor at Cl
                          inlet valve
data field[14..15]      = right justified 12 bit val of thermistor at flush
                          flowmeter in I-branch
data field[16..17]      = right justified 12 bit val of thermistor at  I
                          flowmeter
data field[18..19]      = right justified 12 bit val of thermistor at
                          flange
data field[20..21]      = right justified 12 bit val of thermistor at I
                          inlet valve
data field[22..23]      = right justified 12 bit val of thermistor at
                          ambient air up
data field[24..25]      = right justified 12 bit val of thermistor at
                          ambient air down
```



```
data field[26..27]      = right justified 12 bit val of thermistor at
                          motor body
data field[28..29]      = right justified 12 bit val of thermistor at
                          motor control
data field[30..31]      = right justified 12 bit val of lower R reference
data field[32..33]      = right justified 12 bit val of upper R reference
data field[34..35]      = right justified 12 bit val of internal pressure
data field[36..37]      = right justified 12 bit val of pressurized air
                           pressure
data field[38..39]      = right justified 12 bit val of Cl inlet valve position
data field[40..41]      = right justified 12 bit val of Cl outlet valve position
data field[42..43]      = right justified 12 bit val of Cl flow
data field[44..45]      = right justified 12 bit val of low Cl pressure
data field[46..47]      = right justified 12 bit val of high Cl pressure
data field[48..49]      = right justified 12 bit val of I inlet valve position
data field[50..51]      = right justified 12 bit val of I outlet valve position
data field[52..53]      = right justified 12 bit val of I flow
data field[54..55]      = right justified 12 bit val of low I pressure
data field[56..57]      = right justified 12 bit val of high I pressure
data field[58..59]      = right justified 12 bit val of flush valve position
data field[60..61]      = right justified 12 bit val of flush flow in Cl
                          branch
data field[62..63]      = dummy
data field[64..65]      = right justified 12 bit val of flush flow in I
                          branch
data field[66..67]      = right justified 12 bit val of voltage in discharge
data field[68..69]      = right justified 12 bit val of current in discharge
data field[70..71]      = right justified 12 bit val of voltage in lamp
data field[72..73]      = right justified 12 bit val of current in lamp
data field[74..75]      = right justified 12 bit val of motor speed
data field[76..77]      = right justified 12 bit val of motor current
data field[78..79]      = right justified 12 bit val of thermistor at
                          motor
data field[80..81]      = right justified 12 bit val of voltage at motor
                          battery

total packet length     = 8 + 82 + 2 = 92 bytes

packet duration         = 92 bytes / 960 bytes.sec-1
                        = 95.8 msec @ 9600 baud

                        = 92 bytes / 480 bytes.sec-1
                        = 191.6 msec @ 4800 baud
```

```
############################################################
#
# MACSIMS ON BOARD SOFTWARE
#
# type    : manual pages
# host    : osiris
# file    : /users/proj/Macsims/man/packets/tmpacketinfo
# version : flight 1995
#
############################################################
```

NAME

tmpacketinfo - generic packet structure description

DESCRIPTION

Packet header	: 8 bytes
byte 0	: packet ID (0..255)
byte 1	: packet specific flags
bytes 2 and 3	: packet specific 16 bit packet sequence number
bytes 4 to 7	: 32 bit time tag (units are 10 msec)
Data field	: packet specific data
CRC field	: 16 bit cyclic redundancy check over all preceeding bytes

Inhoud scripts

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type         : flight scripts November 1995
# host         : wichita
# script/file  : CONTENTS
# version      : flight 1995
#
#####################################################


                    ************************************************
                    *                                              *
                    *       M A C S I M S    NOV-1995               *
                    *                                              *
                    *       C O N T E N T S   O F   S C R I P T S  *
                    *                                              *
                    *       Main On Board Computer                 *
                    *                                              *
                    ************************************************

ahkloop
-------
        load name : mobc_ahkloop
        parameters : timedelay
        description :
                endless loop taking ambient housekeeping every
                "timedelay ($1)" seconds.

thkloop
-------
        load name : mobc_thkloop
        parameters : timedelay
        description :
                endless loop taking technological housekeeping every
                "timedelay ($1)" seconds.

hskloop
-------
        load name : mobc_hskloop
        parameters : timedelay
        description :
                endless loop taking ambient, technological and
                chemical housekeeping with a "timedelay ($1)" seconds
                between each package.

itvscancl
---------
        load name : mobc_itvscancl
        parameters :  flow Cl , iiv Cl
        description :
                flow Cl ($1) , iiv XCl ($2) are set.
                Cl source is activated and I source deactivated.
                The lamp is set off.
                Flush flows and ion source HV are as set in global
                variables ($XCL,$XI and $DISCL).
                A chemical housekeeping is taken.
                A loop is executed 5 times with a different value of
                   itv : 950 750 513 348 238.
                itv is set to its value.
                A spectrum is taken with $NRACC accumulations.
                After the loop a status of T4/T6 is taken.
                The discharge ion source is set off.

---------
        load name : mobc_itvscani
        parameters :  flow I , iiv I
        description :
                flow I ($1) , iiv I ($2) are set.
                I source is activated and Cl source deactivated.
                The lamp is set off.
                Flush flows and ion source HV are as set in global
                variables ($XCL,$XI and $DISI).
                A chemical housekeeping is taken.
                A loop is executed 5 times with a different value of
                   itv : 950 750 512 348 238.
                itv is set to its value
                A spectrum is taken with $NRACC accumulations
                After the loop a status of T4/T6 is taken.
                The discharge ion source is set off.

itvscanla
---------
        load name : mobc_itvscanla
        parameters :  iiv lamp
        description :
                iiv lamp ($1) is set.
                I source is deactivated and Cl source deactivated
                The lamp is set on.
                Flush flows and lamp HV are as set in global
                variables ($XCL,$XI and $LAMP).
                A chemical housekeeping is taken.
                A loop is executed 5 times with a different value of
                   itv : 950 750 512 348 238.
                itv is set to its value.
                A spectrum is taken with $NRACC accumulations.
                After the loop a status of T4/T6 is taken.
                The lamp is set off.


lowdac
------
        load_name : mobc_lowdac
        parameters :
        description :
                Putting detector voltages to low values in 2 steps:
                First step  :  rf : 100 , pav : 1000, cpv : 1200, psv : 1500.
                Second step :  pav :  800, cpv :  800, psv :  800, itv : 500.
                Afterwards all voltages are measured (script showvolt).
```

mcalibcli
---

load_name : mobc_mcalibcl
parameters : itvstep, itvstart, itvstop, itv referentie, flow Cl, flow I
description :
Calibration of the masses through an itv-scan.
Installation of the CFMP for Cl and I, with flow Cl ($5) and flow I ($6)
Take a spectrum every itvstep volts, starting at itv=itvstart until itv=itvstop. After each spectrum take another spectrum with itv=itv referentie.

offdac
---

load name : mobc_offdac
parameters :
description :
Putting off rf. Putting off the high voltages.
Afterwards high voltages are measured in order to check if they went to ground.

setdac1
---

load name : mobc_setdac1
parameters : iiv , opb , rf
description :
hvgen on
Setting hv- and rf/dc-values: iiv , opb and rf.
The high voltages are set in two steps to low values.
First step : itv : 500, pav : 500, cpv : 500, psv : 500.
Second step : itv : 800, pav : 800, cpv : 900, psv : 900.

setdac2
---

load name : mobc_setdac2
parameters : iiv , opb , rf
description :
Setting hv- and rf/dc-values: iiv , opb and rf.
The high voltages are set in two steps to nominal values.
First step : itv : 950, pav : 1000, cpv : 1200, psv : 1300.
Second step : cpv : 1500, psv : 1900.
All hv- and rf/dc-values are measured (script showvolt).

showvolt
---

load name : mobc_showvolt
parameters :
description :
Script shows hv rf/dc values.

shrc
---

load name : mobc_shrc
parameters :
description :
Script executed at every system start.
Global variables are set, rf is set to closed loop, discrilevel is set to global variable DISCRI,
hv- and rf/dc-values are set to the raw value corresponding to the lowest possible real voltage.
Gauge and mks100 are put on.
Valve status is checked and the user is asked to set the date (date format : DDD HH MM SS).
Take a technological and an ambient housekeeping.

thkloop
---

load name : mobc_thkloop
parameters : timedelay
description :
Endless loop taking technological housekeeping every "timedelay ($1)" seconds.

tofcl
---

load name : mobc_tofcl
parameters : aantal tof accumulaties, flow Cl, iiv
expected execution time : 1 min.
description :
Use Cl to do a tof measurement
1. current (continuous mode)
2. check of saturation (pulsed one accumulation)
3. time of flight

tofi
---

load name : mobc_tofi
parameters : aantal tof accumulaties, flow I, iiv
expected execution time : 1 min.
description :
Use I to do a tof measurement
1. current (continuous mode)
2. check of saturation (pulsed one accumulation)
3. time of flight

```
-----
        load name : mobc_total
        parameters : flow Cl, iiv Cl, flow I, iiv I, iiv lamp
        expected execution time : 19 min.
        description :
                Execute 3 loops with in each loop :
                        -itvscani
                        -itvscancl
                        -2 dummy spectra
                        -itvscanla

disscan
-------
        load name : mobc_disscan
        parameters : flow Cl, iiv Cl, flow I, iiv I
        expected execution time : 23 min.
        description :
                Execute 5 loops with in each loop :
                        -itvscani
                        -itvscancl
```

```
*****************************************************
*                                                   *
*       M A C S I M S    SEP-1995                   *
*                                                   *
*       C O N T E N T S   O F   S C R I P T S       *
*                                                   *
*       Chemical Factory Processor                  *
*                                                   *
*****************************************************

chkloop
-------
        load name : cfmp_thkloop
        parameters : timedelay
        description :
                Endless loop taking chemical housekeeping every
                "timedelay ($1)" seconds.


flowon
------
        load name : cfmp_flowon
        parameters :
        description :
                 Put the flowmeters on.

shrc
----
        load name :  cfmp_shrc
        parameters :
        description :
                Script executed at every system start.
                Put the manometers on.
                Take a chemical housekeeping.
                Ask the user to set the date.


turbostart
----------
        load name : cfmp_turbostart
        parameters :
        description :
                Start the turbo pump in 3 steps
                Take a chemical housekeeping after each step.

turbodown
---------
        load name : cfmp_turbodown
        parameters :
        description :
                Set the turbo pump to speed 500
                Take a chemical housekeeping.
```

```
turbooff
--------
        load name : cfmp_turbooff
        parameters :
        description :
                Put turbo off

flowoff
-------
        load name : cfmp_flowoff
        parameters :
        description :
                Put the flowmeters off.

setup
-----
        load_name : cfmp_setup
        parameters :
        description :
                Put valves Cl and I in, X open
                Put Cl and I flows to 30 sccm
                Put X flows to global variables $XCL and $XI.
```

Scripts

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_chkloop
# version       : flight 1995
#
#######################################################


remsh insecure
remsh rm chkloop
remsh cat > chkloop
remsh #
remsh # chemical housekeeping
remsh # date : 14/02/94
remsh # parameter : timedelay
remsh #             10 msec units
remsh #
remsh echo start of script chkloop
remsh #
remsh L=0
remsh while test $L -ne 1
remsh do
remsh chk -tm
remsh sleep $1
remsh done
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_flowoff
# version       : flight 1995
#
#####################################################


remsh insecure
remsh rm flowoff
remsh cat > flowoff
remsh #
remsh # date : 21/10/94
remsh #
remsh dis 0
remsh dis
remsh dis off
remsh #
remsh lamp 0
remsh lamp
remsh lamp off
remsh #
remsh flow Cl 0
remsh flow Cl off
remsh #
remsh flow I 0
remsh flow I off
remsh #
remsh flow XCl 0
remsh flow XCl off
remsh #
remsh flow XI 0
remsh flow XI off
remsh #
remsh valve X close
remsh valve I close
remsh valve Cl close
remsh #
remsh pres X off
remsh pres Cl off
remsh pres I off
remsh #
remsh chk -tm
remsh echo flowoff script executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_flowon
# version       : flight 1995
#
#####################################################



remsh insecure
remsh rm flowon
remsh cat > flowon
remsh #
remsh # date : 20/10/94
remsh #
remsh #
remsh flow Cl -r 0
remsh flow Cl on
remsh flow Cl -pa
remsh flow I -r 0
remsh flow I on
remsh flow I -pa
remsh flow XCl -r 0
remsh flow XCl on
remsh flow XCl -pa
remsh flow XI -r 0
remsh flow XI on
remsh flow XI -pa
remsh XCL=50
remsh XI=50
remsh LAMP=1000
remsh DISI=800
remsh DISCL=1500
remsh set
remsh echo flowon script executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_setup
# version       : flight 1995
#
#####################################################


remsh insecure
remsh rm setup
remsh cat > setup
remsh #
remsh # CFMP script setup : initialize valves and flows
remsh # date : 10/10/95
remsh #
remsh flow Cl 30
remsh #
remsh flow I 30
remsh #
remsh flow XCl $XCL
remsh #
remsh flow XI $XI
remsh #
remsh valve X open
remsh valve I in
remsh valve Cl in
remsh #
remsh chk -tm
remsh echo setup script executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_shrc
# version       : flight 1995
#
#######################################################


remsh insecure
remsh rm shrc
remsh cat > shrc
remsh #
remsh # CFMP script "shell run commands"
remsh # date : 10/10/95
remsh #
remsh #
remsh echo start of shrc script of the CFMP
remsh #
remsh pres Cl on
remsh pres I on
remsh pres X on
remsh #
remsh dis -r 1
remsh dis cont
remsh lamp -r 1
remsh lamp cont
remsh sleep 1000
remsh #
remsh chk -tm
remsh echo SET DATE PLEASE
remsh echo script shrc executed
remsh #
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_turbodown
# version       : flight 1995
#
#######################################################


remsh insecure
remsh rm turbodown
remsh cat > turbodown
remsh #
remsh # cfmp script to set the turbo pump to low speed
remsh # date : 21/04/94
remsh #
remsh echo start of turbodown script
remsh #
remsh chk -tm
remsh turbof 1000
remsh sleep 1000
remsh turbof
remsh #
remsh chk -tm
remsh turbof 500
remsh sleep 1500
remsh turbof
remsh #
remsh chk -tm
remsh echo turbodown executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_turbooff
# version       : flight 1995
#
#####################################################


remsh insecure
remsh rm turbooff
remsh cat > turbooff
remsh #
remsh # cfmp script to set the turbo pump off
remsh # date : 21/10/94
remsh #
remsh echo start of turbooff script
remsh #
remsh chk -tm
remsh turbof 0
remsh sleep 1000
remsh turbof
remsh turbof off
remsh #
remsh chk -tm
remsh echo turbooff executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : cfmp_turbostart
# version       : flight 1995
#
#########################################################


remsh insecure
remsh rm turbostart
remsh cat > turbostart
remsh #
remsh # cfmp script for startingthe turbo pump
remsh # date : 28/03/94
remsh #
remsh echo start of turbostart script
remsh #
remsh turboi -r 3900
remsh turbof -r 1
remsh turbof on
remsh #
remsh chk -tm
remsh chk -tm
remsh turbof 500
remsh sleep 1000
remsh turbof
remsh #
remsh chk -tm
remsh turbof 1000
remsh sleep 1500
remsh turbof
remsh #
remsh chk -tm
remsh turbof 1500
remsh sleep 2000
remsh turbof
remsh #
remsh chk -tm
remsh echo turbostart executed
remsh .
remsh secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_ahkloop
# version       : flight 1995
#
#####################################################


insecure
rm ahkloop
cat > ahkloop
#
# ambient housekeeping
# date : 14/02/94
# parameter : timedelay
#             10 msec units
#
echo start of script ahkloop
#
M=0
while test $M -ne 1
do
ahk -tm
sleep $1
done
echo script ahkloop executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_disscan
# version       : flight 1995
#
#####################################################


insecure
rm disscan
cat > disscan
#
# MOBC script disscan : Cl and I, no lamp
# date : 10/10/95
# parameters : (1.flow Cl)
#              (2.iiv Cl)
#              (3.flow I)
#              (4.iiv I)
#
echo start of script disscan
#
cc=0
while test $cc -lt 5
do
date
showvolt
#
remsh dis -r 1
remsh lamp -r 1
#
msp -s
msp -b $NRACC
#
itvscancl $1 $2
#
sleep 200
#
itvscani $3 $4
#
sleep 200
#
itv 950
#
# 2 dummy spectra to clean the detector
echo first dummy
msp $NRACC
echo second dummy
msp $NRACC
#
let cc=$cc+1
done
echo script disscan executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_hskloop
# version       : flight 1995
#
#####################################################


insecure
rm hskloop
cat > hskloop
#
# ambient, technological and chemical  housekeeping
# date : 23/03/94
# parameter : timedelay
#             10 msec units between each housekeeping
#
echo start of script hskloop
#
K=0
while test $K -ne 1
do
ahk -tm
sleep $1
thk -tm
sleep $1
remsh chk -tm
sleep $1
done
#
echo script hskloop executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_itvscancl
# version       : flight 1995
#
#######################################################


insecure
rm itvscancl
cat > itvscancl
#
# MOBC script itvscancl : scan 5 standard itv values with Cl
# date : 10/10/95
# parameters : (1.flow Cl)
#              (2.iiv Cl)
#
echo start of script itvscancl
#
iiv $2
iiv
#
remsh valve X open
remsh valve I out
remsh valve Cl in
#
remsh flow Cl $1
remsh flow XI $XI
remsh flow XCl $XCL
#
remsh lamp -r 1
#
remsh dis 0
remsh dis 600
remsh dis 1000
remsh dis $DISCL
#
sleep 200
remsh chk -tm
for xxx  in 950 750 513 348 238
do
itv  $xxx
sleep 100
echo itv :
itv
echo measurement started
msp $NRACC
done
#
msp -s
remsh dis -r 1
remsh valve Cl out
#
remsh chk -tm
echo script itvscancl executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_itvscani
# version       : flight 1995
#
#######################################################


insecure
rm itvscani
cat > itvscani
#
# MOBC script itvscani : scan 5 standard itv values with I
# date : 10/10/95
# parameters : (1.flow I)
#              (2.iiv I)
#
echo start of script itvscani
#
iiv $2
iiv
#
remsh valve X open
remsh valve Cl out
remsh valve I in
#
remsh flow I $1
remsh flow XCl $XCL
remsh flow XI $XI
#
remsh lamp -r 1
#
remsh dis 0
remsh dis 600
remsh dis $DISI
#
sleep 200
remsh chk -tm
for xxx  in 950 750 513 348 238
do
itv  $xxx
sleep 100
echo itv :
itv
echo measurement started
msp $NRACC
done
#
msp -s
remsh dis -r 1
remsh valve I out
#
remsh chk -tm
echo script itvscani executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_itvscanla
# version       : flight 1995
#
#####################################################


insecure
rm itvscanla
cat > itvscanla
#
# MOBC script itvscanla : scan 5 standard itv values with lamp
# date : 10/10/95
# parameter : (1.iiv lamp)
#
echo start of script itvscanla
#
iiv $1
iiv
#
remsh valve X open
remsh valve I out
remsh valve Cl out
#
remsh flow XCl $XCL
remsh flow XI $XI
#
remsh dis -r 1
#
remsh lamp 0
remsh lamp 600
remsh lamp 1200
remsh lamp $LAMP
#
remsh chk -tm
for xxx  in 950 750 513 348 238
do
itv  $xxx
sleep 100
echo itv :
itv
echo measurement started
msp $NRACC
done
#
msp -s
remsh lamp -r 1
#
remsh chk -tm
echo script itvscanla executed
.
secure
```

```
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_lowdac
# version       : flight 1995
#
#####################################################


insecure
rm lowdac
cat > lowdac
#
#  Mobc script lowdac for flight
# date : 23/03/94
#
echo start of script lowdac
#
date
pav 1000
cpv 1200
psv 1500
rf 100
sleep 200
showvolt
pav 800
cpv 800
psv 800
itv 500
sleep 200
showvolt
echo script lowdac executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_mcalibcli
# version       : flight 1995
#
#####################################################


insecure
rm mcalibcli
cat > mcalibcli
#
# MOBC script mcalibcli for itv scan with I and Cl
# date : 10/10/95
# parameters : (1.itvstep)
#              (2.itvstart)
#              (3.itvstop)
#              (4.referentie itv)
#              (5.flow Cl)
#              (6.flow I)
#
echo start of script mcalibcli
#
remsh valve X open
remsh valve Cl in
remsh valve I in
#
remsh flow Cl $5
remsh flow I $6
remsh flow XI $XI
remsh flow XCl $XCL
#
remsh lamp -r 1
#
msp -b $NRACC
#
remsh dis 0
remsh dis 600
remsh dis 1000
remsh dis $DISCL
#
sleep 200
remsh chk -tm
#
itv $2
sleep 100
showvolt
#
ITV=0
let ITV=$ITV + $2
while test $ITV -le $3
do
#
date
#
echo measurement started
#
sleep 100
itv  $ITV
echo itv :
sleep 100
msp  NRACC
#
sleep 100
itv $4
echo itv :
sleep 100
itv
msp $NRACC
#
let ITV=$ITV + $1
done
#
msp -s
#
remsh dis -r 1
#
remsh valve Cl out
remsh valve I out
#
echo script mcalibcli executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_offdac
# version       : flight 1995
#
#####################################################


insecure
rm offdac
cat > offdac
#
#  Mobc script offdac for flight
#  before set off the instrument
# date : 23/03/94
#
echo start of script offdac
#
date
itv 500
pav 800
cpv 1000
psv 1000
sleep 200
showvolt
itv off
pav off
cpv off
psv off
hvgen off
sleep 1000
iiv 0
opb 0
grid 0
rf 0
rf off
showvolt
echo script offdac executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_restart
# version       : flight 1995
#
#####################################################


insecure
rm restart
cat > restart
#
# MOBC script restart : execute it after break down
# date : 10/10/95
#
echo start of script restart
#
gauge on
mks100 on
sleep 100
#
ahk -tm
sleep 200
thk -tm
sleep 200
remsh chk -tm
sleep 200
#
tof -r 100
tof -d 10000
tof -p 1000
tof -g 1 1
#
echo CHECK PRESSURE,
echo SET DATE AGAIN
echo DATE FORMAT : DDD HH MM SS
echo EXECUTE SCRIPT SETDAC1
echo REINSTALL CHEMICAL FACTORY
echo EXECUTE SCRIPT REMSH FLOWON
echo EXECUTE SCRIPT REMSH TURBOSTART
#
echo script restart executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_setdac1
# version       : flight 1995
#
#####################################################


insecure
rm setdac1
cat > setdac1
#
#  Mobc script setdac1 ,low setting of voltages
# date : 23/03/94
# parameters: iiv , opb , rf
#
echo start of script setdac1
#
rf on
iiv $1
opb $2
rf $3
hvgen on
itv on
pav on
cpv on
psv on
itv 500
pav 500
cpv 500
psv 500
sleep 1000
showvolt
itv 800
pav 800
cpv 900
psv 900
sleep 1000
showvolt
echo script setdac1 executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_setdac2
# version       : flight 1995
#
#####################################################


insecure
rm setdac2
cat > setdac2
#
#  Mobc script setdac2 ,nominal setting of voltages
# date : 23/03/94
# parameters: iiv , opb , rf
#
echo start of script setdac2
#
iiv $1
opb $2
rf $3
itv 950
pav 1000
cpv 1200
psv 1300
sleep 500
showvolt
cpv 1500
psv 1600
sleep 300
showvolt
psv 1900
showvolt
echo script setdac2 executed
.
secure
```

```
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_showvolt
# version       : flight 1995
#
#####################################################


insecure
rm showvolt
cat > showvolt
#
#  Mobc script showvolt
# date : 22/02/94
#
echo start of script showvolt
#
echo ITV is
itv
echo PAV is
pav
echo CPV is
cpv
echo PSV is
psv
echo IIV is
iiv
echo OPB is
opb
echo RF is
rf
echo IONGATE is
grid
echo script showvolt executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_shrc
# version       : flight 1995
#
#####################################################


insecure
rm shrc
cat > shrc
#
# MOBC script : "shell run commands"
# date : 10/10/95
#
echo start of shrc script
#
cll
itv -r 1
itv -pa
pav -r 1
pav -pa
cpv -r 1
cpv -pa
psv -r 1
psv -pa
iiv -r 1
iiv -pa
opb -r 1
opb -pa
rf -r 1
rf -pa
grid -r 1
grid -pa
#
gauge on
mks100 on
#
sleep 1000
NRACC=500
DISCRI=5
msp -l $DISCRI
echo CHECK IF VALVE OPENING IS CORRECT
valve status
echo SET DATE PLEASE
echo date format : DDD HH MM SS
#
tof -r 100
tof -d 10000
tof -p 1000
tof -g 1 1
#
ahk -tm
thk -tm
#
echo script shrc executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_thkloop
# version       : flight 1995
#
#####################################################


insecure
rm thkloop
cat > thkloop
#
# technological housekeeping
# date : 14/02/94
# parameter : timedelay
#             10 msec units
#
echo start of script thkloop
#
L=0
while test $L -ne 1
do
thk -tm
sleep $1
done
#
echo script thkloop executed
.
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_tofcl
# version       : flight 1995
#
#####################################################



insecure
rm tofcl
cat > tofcl
#
# MOBC script tofcl : time of flight measurement with Cl
# date : 10/10/95
# parameters : (1.tofacc)
#              (2.flow Cl)
#              (3.iiv)
#
echo start of script tofcl
#
iiv $3
iiv
#
remsh valve X open
remsh valve I out
remsh valve Cl in
#
remsh flow Cl $2
remsh flow XI $XI
remsh flow XCl $XCL
#
remsh lamp -r 1
#
remsh dis 0
remsh dis 600
remsh dis 1000
remsh dis $DISCL
#
sleep 200
remsh chk -tm
#
echo measurement started
msp -s
#
echo 1. current
tof 1
#
sleep 200
remsh dis pulse
sleep 200
remsh chk -tm
echo 2. check saturation
tof 1
#
sleep 200
echo 3. time of flight
tof $1
#
remsh dis -r 1
remsh dis cont
#
echo script  o cl execu e
.
secure
```

```
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_tofi
# version       : flight 1995
#
#########################################################



insecure
rm tofi
cat > tofi
#
# MOBC script tofi : time of flight measurement with I
# date : 10/10/95
# parameters : (1.tofacc)
#              (2.flow I)
#              (3.iiv)
#
echo start of script tofi
#
iiv $3
iiv
#
remsh valve X open
remsh valve Cl out
remsh valve I in
#
remsh flow I $2
remsh flow XI $XI
remsh flow XCl $XCL
#
remsh lamp -r 1
#
remsh dis 0
remsh dis 600
remsh dis $DISI
#
sleep 200
remsh chk -tm
#
echo measurement started
msp -s
#
echo 1. current
tof 1
#
sleep 200
remsh dis pulse
sleep 200
remsh chk -tm
echo 2. check saturation
tof 1
sleep 200
#
echo 3. time of flight
tof $1
#
remsh dis -r 1
remsh dis cont
#
remsh chk -tm
```

```
secure
```

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : mobc_total
# version       : flight 1995
#
#####################################################


insecure
rm total
cat > total
#
# MOBC script total : Cl, I and lamp
# date : 10/10/95
# parameters : (1.flow Cl)
#              (2.iiv Cl)
#              (3.flow I)
#              (4.iiv I)
#              (5.iiv lamp)
#
echo start of script total
#
cc=0
while test $cc -lt 3
do
date
showvolt
#
remsh dis -r 1
#
remsh lamp -r 1
#
msp -s
msp -b $NRACC
#
itvscani $3 $4
#
#sleep 200
#
itvscancl $1 $2
#
sleep 200
#
itv 950
#
# 2 dummy spectra to clean the detector
echo first dummy
msp $NRACC
echo second dummy
msp $NRACC
#
itvscanla $5
#
sleep 200
#
let cc=$cc+1
done
echo script total executed
.
secure
```

Scenario

```
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : SCENARIO
# version       : flight 1995
#
#####################################################
```

```
                    ************************************************
                    *                                              *
                    *      F L I G H T   S C E N A R I O           *
                    *                                              *
                    *      Macsims nov 1995                        *
                    *                                              *
                    ************************************************
```

I. ON LAUNCH PAD
---

1. putting on the instrument (script "shrc" is executed automatically)

   in Tc_onoff window execute "MOBC on"
                               "execute"
                               "CFMP on"
                               "execute"
                               "Octopole on"
                               "execute"
                               "Turbo on"
                               "execute"

2. setting the date (date format DDD HH MM SS)

   "Set date" push button in Tc_text window

   Write down the exact time difference between the instrument time
   and the CNES time.

   Optional : adjust the instrument manually to the CNES-time using
              the "date DDD HH MM SS" command.

3. check on TCecho :

   if valve is opened (valve status is asked in "shrc")

   check on Telemetry :

   if thk,ahk and chk were transmitted.

      if they were not transmitted execute "ahk -tm " or
                                            "thk -tm" or
                                            "remsh chk -tm"

      if still no response, verify telemetry connections !!!!!

   what the pressure is at the gauge (expected value : 10 exp -7 bar)
                                                  (expected value : <1 mbar)

   what the pap pressure is (expected value : 6 -> 8 bar)

   what is : Cl, I low pressure (expected value : 2 bar)
             Cl, I high pressure (expected value : 100 bar)
             Flush low pressure (expected value : 2 bar)
             Flush high pressure (expected value : 200 bar)

   remark : T° at turbine motor is not valid

4. execute "remsh flowon" and ask for housekeeping ("remsh chk -tm"):

   what are Cl flow,I flow, flush flows (expected value : 0 -> 1370 sccm)

   verify GV's in MOBC ("set"): NRACC=500
                                DISCRI=5

   verify GV's in CFMP ("remsh set"): XCL=50
                                      XI=50
                                      LAMP=1000
                                      DISI=800
                                      DISCL=1500

   modify GV's if necessary

5. check if mks100 is still on

6. execute "setdac1 5 0.5 200" (iiv opb rf)

   check on TC echo the showvolt result or measure the high voltages

   if rf = 0 then execute : "cll on"
                             "rf 1"
                             "rf on"
                             "rf 200"
                             "rf"
                             test again !!!

7. execute "msp -s"
            "msp -b 500"
            "msp 500"

   check on Telemetry if T4 is working
```
```

8. when Piet is ready : test the opening system:

With the light bulb connected

execute "cap test"

execute "thk"

execute "cap enable"
"cap confirm"
"cap open"

After connecting the cables (after Piet's OK)

execute "cap test" (expected value : about 3370 dec)

execute "thk"

check the value of the inlet cap (expected value : closed)
check the value of the outlet cap (expected value : closed)

9. execute "thk -tm" and check if value of gauge is OK
(expected value: 10 exp-7 -> 10 exp -8)

10. if Piet agrees and if gauge value is good :

execute "setdac2 5 0.5 200" (iiv opb rf)

check on TC echo the showvolt results or measure the high voltages
and rf/dc-voltages

10bis. execute "msp -s", "msp -b 500" and "msp 500"

11. take tofspectrum ("tof 1") and check if tof-parameters have their default value. Check other values on the screen.

expected values : rate : 100 us
delay : 10 ms
period : 1 ms
gain : 1 x 1

********** NOW MOBC is ready **********

12. execute "remsh turbostart"

check if the T° at turbine motor is valid
expected value : between 30 and 40 °C

check on telemetry (chk) window :
if motor speed is 1500 rps
if motor current is about 1 A

13. execute "remsh dis cont"
"remsh dis 1000"
"remsh dis" (verify if dis is approx. 1000 V)
(no current is expected)
"remsh chk -tm"

execute "remsh dis 400 "
"remsh dis" (expected value : approx. 400 V)
(no current is expected)

14. execute "remsh lamp cont"
"remsh lamp 1200"
"remsh lamp" (verify if lamp is 1200 V)
(a current of approx. 550 µA is expected)
"remsh chk -tm"

check on telemetry (chk) window if current is approx. 550 µA

"remsh lamp 400"
"remsh lamp" (expected value : approx. 380 V)
(expected current : approx. ... µA)

15. execute "lowdac"

16. check functioning of gas valves :

execute "remsh valve X open"
"remsh chk -tm"
"remsh valve X close"
"remsh chk -tm"

execute "remsh valve I in"
"remsh chk -tm"
"remsh valve I close"
"remsh chk -tm"

execute "remsh valve Cl in"
"remsh chk -tm"
"remsh valve Cl close"
"remsh chk -tm "

"remsh turbooff"

********** NOW CFMP is ready **********

18. execute "hskloop 1000 &" (backgroundtask in MOBC)

    check if every 10 seconds a housekeeping packet arrives (consecutively ambient, technological and chemical housekeeping)

    in order to finish this task :

        execute "kill 12"

19. just before launch : set Cl and I flows and valves to "out" :

        execute "remsh flow Cl 50"
                "remsh flow I 50"
                "remsh valve Cl out"
                "remsh valve I out"

********** NOW MACSIMS is ready for launch **********

II. DURING ASCENT

20. 5 to 10 minutes after take off : start the turbine again :
        "remsh turbostart"
        check on TM (chk) if turbine is starting

21. check on TM (chk) the values of discharge voltage (approx. 400 V)
                                    lamp voltage (approx. 380 V)
                                    lamp current (approx. ... µA)

    check all the temperatures and pressures

        if Cl, I and/or X valve temperatures are too low :

            execute "remsh valve Cl close"
            check status
            execute "remsh valve Cl out"
            check status

            and/or

            execute "remsh valve I close"
            check status
            execute "remsh valve I out"
            check status

            and/or

            execute "remsh valve X open"
            check status
            execute "remsh valve X close"
            check status

22. Once in a while execute : msp -s (housekeeping T4)

III. ON CEILING ALTITUDE

23. kill the housekeeping tasks to avoid overload of the MOBC :

        execute "kill 12"

24. lower the turbine velocity : "remsh turbodown"

25. execute "valve status"

26. opening of the shutter : "cap enable"
                             "cap confirm"
                             "cap open"
                             "cap disable"

                             "thk"
                             check cap status

                             "ahk -tm"
                             check mks100 pressure (expected value : 13 mbar)

27. restart the housekeeping loop

        execute "hskloop 1000 &"

28. execute "setdac1 5 0.5 200" (iiv opb rf)
    (setting hv and rf/dc to low values and putting on rf and high voltages)

29. execute "remsh turbostart"

30. execute "setdac2 5 0.5 200" (iiv opb rf)
    (setting hv and rf/dc to standard values)

31. execute "remsh dis -r 1"
    "remsh lamp -r 1"
    "msp -b 500"
    "msp -b 500"
    check if background is OK

31bis. execute "itvscanla 5" (iiv lamp)

32. mass calibration

    execute "mcalibcli 50 200 1100 950 100 25"
    (itvstep,itvstart,itvstop,itvref,flow Cl,flow I)

    expected execution time : 12 min. (with this itvstep=50)

33. searching for ideal parameters :

    execute "itvscancl 100 5" (flow Cl, iiv Cl)

    if necessary :
    change script parameters and reexecute until OK
    change GV's and reexecute until OK

    execute "itvscani 50 5" (flow I, iiv I)

    if necessary :
    change script parameters and reexecute until OK
    change GV's and reexecute until OK

    execute "itvscanla 5" (iiv lamp)
    if necessary :
    change script parameters and reexecute until OK
    change GV's and reexecute until OK

34. start measurements using the found parameters :

    execute "total 100 5 50 5 5"
    (flow Cl, iiv Cl, flow I, iiv I, iiv lamp)

    expected execution time : 19 min.

    to stop this script : Ctrl_c after the execution of "itvscanla"

35. after every total-script take a time-of-flight-measurement :

    execute "tofcl 100 100 5" or "tofi 100 100 5"
    (#acc,Cl flow,iiv) (#acc,I flow,iiv)

    expected execution time : 1 min. per script

IV. AT APPROX. 40 to 50 MBAR
---

36. mass calibration

    execute "mcalibcli 50 200 1100 950 100 25"
    (itvstep,itvstart,itvstop,itvref,flow Cl,flow I)

    expected execution time : 12 min. (with this itvstep=50)

V. END OF USEFUL MEASUREMENTS
---

37. execute "lowdac" and "offdac" (putting off high voltages and rf)

38. close valve (grid on + valve close + valve status + grid off)

39. execute "remsh dis 0"
    "remsh dis"
    "remsh dis off"
    "remsh lamp 0"
    "remsh lamp"
    "remsh lamp off"
    "remsh valve X open"
    "remsh valve Cl out"
    "remsh valve I out"
    "remsh flow XI 1370"
    "remsh flow XCl 1370"
    "remsh flow I 1370"
    "remsh flow Cl 1370"

VI. BEFORE SQUIBING
---

40. when bottles are "empty" execute

    execute "remsh turbodown"
    "remsh turbooff"
    "remsh flowoff"

```
Turbo off
execute
octopole off
execute
CFMP off
execute
MOBC off
execute
```

```
*************************************************
*                                               *
*  ALTERNATIVES FOR THE MEASUREMENT             *
*                                               *
*************************************************
```

I.If lamp problem:

execute "disscan 100 5 50 1" (flow Cl, iiv Cl, flow I, iiv I)

expected execution time : 23 min.

II. If just Cl scan :

execute "itvscancl 100 5" (flow Cl, iiv Cl)

III. If just I scan :

execute "itvscani 50 5" (flow I, iiv I)

IV. If just lamp scan :

execute "itvscanla 5" (iiv lamp)

V. Changing detector high voltages, e.g. cpv to 1600, 1700 Volts

VI. For tof measurements use tofcl- and tofi-scripts

execute "tofcl 100 100 5" and/or "tofi 100 100 5"

```
************************************************
*                                              *
*  TROUBLESHOOTING AND EMERGENCY INTERVENTION   *
*                                              *
************************************************
```

If the program crashes (no TC possible) :

I. switch off the instument :

in Tc_onoff window execute

```
"Turbo off"
"execute"
"Octopole off"
"execute"
"CFMP off"
"execute"
"MOBC off"
"execute"
```

II. wait at least 10 seconds.

III. switch on the instrument again :

in Tc_onoff window execute

```
"MOBC on"
"execute"
"CFMP on"
"execute"
"Octopole on"
"execute"
"Turbo on"
"execute"
```

IV. Restitute the situation before breakdown

run through the next points of the scenario :

- point 2 : date procedure
- point 3 : check execution of shrc - check pressures
- extra 1 : check presence of scripts in MOBC ("dir")
- extra 2 : check presence of scripts in CFMP ("remsh dir")
- point 4 : check flows - check global variables
- point 5 : check pressure in flow tube
- point 27 : restart housekeeping loop
- point 28 : set detector voltages - first step
- point 29 : start and check turbine
- point 30 : set detector voltages to nominal values
- extra 3 : check discharge

```
"remsh dis cont"
"remsh dis 1000"
"remsh dis" + check
check TM
"remsh dis -r 1"
```

- extra 4 : check lamp

```
"remsh lamp cont"
"remsh lamp 1200"
"remsh lamp" + check
check TM
"remsh lamp -r 1"
```

- extra 5 : check T4 "msp -s"
- extra 6 : take a background "msp -b 500" + check
- extra 7 : install non standard parameters if necessary
- point 34 : start measurements again

Controlelijst

```
#
# MACSIMS ON BOARD SOFTWARE
#
# type          : flight scripts November 1995
# host          : wichita
# script/file   : CHECKLIST
# version       : flight 1995
#
#####################################################
```

```
***********************************************
*                                             *
* MACSIMS - LEON - NOVEMBER 1995 - CHECK LIST *
*                                             *
***********************************************
```

A C T I O N C H E C K E D

Before putting the container on MOBC

---

check if all bottom flange holes are closed ..... at .....

check if the big O-ring has been cleaned ..... at .....

check if all critical parts have been thermally isolated :

T4-detector ..... at .....

electrometer ..... at .....

put silicagel on all levels (... boxes) ..... at .....

check if all internal thermistors are in place ..... at .....

check if batteries are connected (MOBC and octopole supply) . at .....

check battery voltages on rack through test connector
MOBC-J01 :

MOBC-supply : ..... Volt ..... at .....

Octopole supply : ..... Volt ..... at .....

check if battery connectors are torqued ..... at .....

check if absolute pressure valves for helium have red or
alu O-rings ..... at .....

check if isolation screen has been installed ..... at .....

t e iso ation screen ..... at .....

check if the big O-ring stays well placed while putting
the container (use pocket lamp to verify) ..... at .....

container installed ..... at .....

Before putting the container on CFMP

---

check if all bottom flange holes are closed ..... at .....

check if the big O-ring has been cleaned ..... at .....

check if connectors are on flow meters ..... at .....

check if all critical parts have been thermally isolated :

flowmeters ..... at .....

bottom flange ..... at .....

put silicagel on all levels except the lowest (... boxes) .. at .....

check if all internal thermistors are in place ..... at .....

check if two battery sets are connected (CFMP and turbo) .... at .....

check battery voltages on rack through test connector
CFMP-J01:

CFMP-supply : ..... Volt ..... at .....

Turbine supply : ..... Volt ..... at .....

check if the battery connectors have been torqued ..... at .....

check if gas bottles are opened (... bottles) ..... at .....

check if isolation screen has been installed ..... at .....

check if the big O-ring is still clean after installing
the isolation screen ..... at .....

check if the big O-ring stays well placed while putting
the container (use pocket lamp to verify) ..... at .....

container installed ..... at .....

After putting the container on MOBC

check if all container screws have been torqued ..... at .....

check if all external thermistors (4) are in place ..... at .....

check if the cable between MOBC (MOBC-J05) and
CFMP (CFMP-J02) is OK ..... at .....

check if connector MOBC-J02 has been cabled
(external thermistors and opening systems) ..... at .....

check if connector MOBC-J03 has been cabled
(telemetry-telecommand interface with CNES) ..... at .....

check if TM/TC-connection at CNES-side is OK ..... at .....

check battery voltages on rack through test connector
MOBC-J01 :

MOBC-supply : ..... Volt ..... at .....

Octopole supply : ..... Volt ..... at .....

check if test cable on MOBC-J01 has been disconnected and
replaced by a strap ..... at .....

double check the mounting of all external flange connectors
(CANNON)..... at .....

check if closing valves cryopumps are torqued
(use torque meter) ..... at .....

check electronics of opening systems with light bulbs ..... at .....

check if two opening systems are connected ..... at .....

check if connector screws of opening systems are torqued .... at .....

helium filling :

small cryopump : started at ..... stopped at : .....

big cryopump : started at ..... stopped at : .....

After putting the container on CFMP

check if all container screws have been torqued ..... at .....

check if closing valves flow tube are torqued
(use torque meter) ..... at .....

check if strap connector has been put on CFMP-valves
connector ..... at .....

check if a protection cap has been put to CFMP-J03
connector ..... at .....

check battery voltages on rack through test connector
CFMP-J01 :

CFMP-supply : ..... Volt ..... at .....

Turbine supply : ..... Volt ..... at .....

check if test cable on CFMP-J01 has been disconnected and
replaced by a protection cap ..... at .....

double check the mounting of all external flange connectors
(CANNON) ..... at .....

check pressure before disconnecting vacuum pumps :

detector : ..... mbar ..... at .....

flow tube : ..... mbar ..... at .....

---

check if helium exhaust tubes are directed towards top of instrument ..... at .....

check if tubes for ion source outlet are directed towards the top of the instrument ..... at .....

check if securing cables are mounted to containers ..... at .....

check if extra protective structure is mounted en screwed ... at .....

check if all external thermistors are free ..... at .....

check if TM/TC-cable is directed towards the top of the instrument (avoid strain on cable) ..... at .....

---

check if power comes from generator ..... at .....

check if all workstations, screens, consoles and mouses are alive ..... at .....

check if network transceiver blocks are locked ..... at .....

check if telemetry and telecommand connectors are locked .... at .....

check if connection to PC-table is alive ..... at .....

follow flight scenario ..... at .....