EUROPEAN
SOUTHERN
OBSERVATORY

# ESO-MIDAS

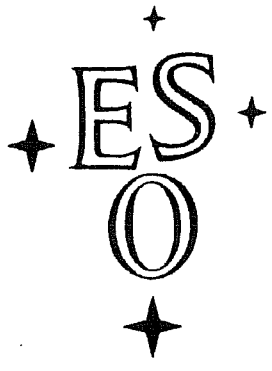
MUNICH
IMAGE
DATA
ANALYSIS
SYSTEM

# VOLUME C: COMMAND DESCRIPTION

ESO OPERATING MANUAL NO. 1 REFERENCE NUMBER: MID-MAN-ESO-11000-0004

# ESO-MIDAS User Guide

## Volume C: Detailed Command Description



MIDAS Release **95NOV**

Reference Number: MID-MAN-ESO-11000-0004

| Section | Title | Date |
|---|---|---|
| Chapter 1 | Introduction | 1-November-1995 |
| Appendix A | MIDAS Control Language | 1-November-1995 |
| Appendix B | Core commands | 1-November-1995 |
| Appendix C | Application commands | 1-November-1995 |
| Appendix D | Standard Reduction: echelle | 1-November-1995 |
| Appendix E | Standard Reduction: ccdred | 1-November-1995 |
| Appendix F | Standard Reduction: ccdtest | 1-November-1995 |
| Appendix G | Standard Reduction: do | 1-November-1995 |
| Appendix H | Standard Reduction: irac2 | 1-November-1995 |
| Appendix I | Standard Reduction: irspec | 1-November-1995 |
| Appendix J | Standard Reduction: long | 1-November-1995 |
| Appendix K | Standard Reduction: optopus | 1-November-1995 |
| Appendix L | Standard Reduction: pisco | 1-November-1995 |
| Appendix M | Standard Reduction: spec | 1-November-1995 |
| Appendix N | Contributed commands: cloud | 1-November-1995 |
| Appendix O | Contributed commands: daophot | 1-November-1995 |
| Appendix P | Contributed commands: esolv | 1-November-1995 |
| Appendix Q | Contributed commands: geotest | 1-November-1995 |
| Appendix R | Contributed commands: invent | 1-November-1995 |
| Appendix S | Contributed commands: mva | 1-November-1995 |
| Appendix T | Contributed commands: pepsys | 1-November-1995 |
| Appendix U | Contributed commands: romafot | 1-November-1995 |
| Appendix V | Contributed commands: surfphot | 1-November-1995 |
| Appendix W | Contributed commands: tsa | 1-November-1995 |
| Appendix X | Contributed commands: astromet | 1-November 1995 |

EUROPEAN SOUTHERN OBSERVATORY
Data Management Division
Karl–Schwarzschild–Straße 2, D–85748 Garching bei München
Federal Republic of Germany

# Contents

# Chapter 1

# Introduction

## 1.1 How to use the MIDAS Manual

This document is intended to be a description of how to use the various facilities available in the MIDAS system. The manual consists of two volumes:

**Volume A:** describes the basic MIDAS system with all general purpose facilities such as MIDAS Control Language, all available commands, data input/output (including plotting and image display), table system (MIDAS Data Base). Site specific features are given in an appendix.

**Volume B:** describes how to use the MIDAS system for astronomical data reduction. Application packages for special types of data or reductions (*e.g.* long slit and echelle spectra, object search, or crowded field photometry) are discussed assuming intensity calibrated data. A set of appendices gives a detailed description of the reduction of raw data from ESO instruments.

**Volume C:** gives the detailed description for all commands available.

It is intended that users will mainly need Volume A for general reference. For specific reduction of raw data and usage of special astronomical packages, Volume B will be more informative. A printed version of the MIDAS help files is available in Volume C. Users are recommended to use the on-line help facility which always gives a full up to date description of the commands available.

### 1.1.1 New Users

To be able to use MIDAS, it is a great advantage to have some basic knowledge of computer systems such as how to login, use of the file editor and simple system commands. Such instructions can normally be found in local system documentation or in Appendix C of Volume A. After having acquired this knowledge, new users should read Chapter 2 Volume A carefully. This will give a basic introduction to the MIDAS system with some examples.

### 1.1.2 Site Specific Features

MIDAS is used at many different sites on a large variety of configurations. The main part of this manual does not refer to special configurations or hardware devices. Site specific implementations and details of the local installation can be found in Appendix C of Volume A.

Requests and questions are acknowledged when received and processed as soon as possible, normally within a few days. Also, users are strongly encouraged to send suggestions and comments via the **MIDAS Hot–Line**.

In urgent cases, users can use a special MIDAS Support telephone service at ESO on the number +49-89-32006-456. This line is connected to the MIDAS Users Support which is able directly to answer questions concerning MIDAS or investigate the problem in more complicated cases. Although this telephone service is available we prefer that questions or requests are submitted in writting via the MIDAS **Hot–Line**. This makes it easier for us to process the requests properly. A database with problem reports and answers is available for interogation using the STARCAT utility at ESO. General information concerning the MIDAS system should be addressed to User Support Group, European Southern Observatory, Karl-Schwarzschild-Straße 2, D-85748 Garching bei München (attn: Resy de Ruijsscher).

Besides these support services, a newsletter, the ESO-MIDAS Courier, is issued twice a year.

## 1.2 Other Relevant Documents

There are several other documents relevant to the MIDAS system. General descriptions of the system can be found in the following references:

- Banse, K., Crane, P. Ounnas, C., Ponz, D., 1983 : 'MIDAS' in *Proc. of DECUS*, Zurich, p.87
- Grosbøl, P., Ponz, D. , 1985 : 'The MIDAS Table File System', *Mem.S.A.It.* **56**, p.429
- Banse, K., Grosbøl, P., Ponz, D., Ounnas, C., Warmels, R., 'The MIDAS Image Processing System in Instrumentation for Ground Based Astronomy: Present and Future, L.B. Robinson, ed., New York, Springer Verlag, p.431

For general bibliographic reference to the MIDAS system (VAX/VMS version), the first reference in the above list should be used.

Detailed technical information of software interfaces and designs used in MIDAS is given in the following documentation:

- MIDAS Environment
- MIDAS IDI–routines
- AGL Reference Manual

Users who want to write their own application programs for MIDAS should read the MIDAS Environment document which gives the relevant information and examples.

For users who have to work with both the IHAP and MIDAS systems a cross–reference document has been made for the most commonly used commands:

- MIDAS-IHAP/IHAP-MIDAS Cross-Reference

The above documents can be obtained by contacting the User Support Group (e.g. via the HOT-LINE).

# Appendix A

# MIDAS Control Language

## BRANCH *core* 15-JAN-1987 KB

**Purpose:** Do multi-way branching.

**Syntax:** `BRANCH var comparisons labels`

| | |
|---|---|
| `var` | name or contents of key (type integer or character) |
| `comparisons` | string of integer or character comparison-values separated by commas |
| `labels` | string of labels (without colon (:) in the end), separated by commas |

**See also:** GOTO
Chapter 3 of MIDAS Users Guide, volume A

**Note:** If var is equal to any of the comparison-values, flow-of-control is transferred to the corresponding label as given in the parameter 'labels'.
If no match is found, the statement following the BRANCH command will be executed.

**Examples:** `BRANCH P1(4:4) A,B,C,D,X LAB_A,LAB_B,LAB_C,LAB_D,LAB_X`
if P1(4:4) is the character C, a jump to the line with the label LAB_C: will be done

`BRANCH inputi(11) 2,04,6,8 L1,L2,L3,L4`
if INPUTI(11) is 4 a jump to the line with label L2: is done

## CONTINUE *core* 09-DEC-1994 KB

**Purpose:** Stop a 'paused' Midas procedure and clear the internal PAUSE structures, i.e. now you can execute another procedure containing a PAUSE command.

**Syntax:** `CONTINUE/CLEAR`

**See also:** CONTINUE, PAUSE

**Note:** Since only one procedure can be in 'paused' state at a time, you have to either resume execution of that procedure or stop it before another procedure can execute a PAUSE command.

**Examples:** `CONTINUE/CLEAR`
Clear the 'paused' state of the procedure which had stopped after a PAUSE command in its code, do NOT continue the execution of that procedure.

## CONTINUE *core* 09-DEC-1994 KB

**Purpose:** Continue the Midas procedure which has been previously interrupted via the PAUSE command.

**Syntax:** `CONTINUE`

**See also:** CONTINUE/CLEAR, PAUSE

**Note:** None

**Examples:** `CONTINUE`
Continue execution of the procedure which had stopped after executing a PAUSE command in its code.

# CROSSREF

| CROSSREF | *core* | 15-JAN-1987 KB |
|---|---|---|

**Purpose:** define cross reference labels for the 8 parameters

**Syntax:** `CROSSREF labl1 labl2 labl3 labl4 labl5 labl6 labl7 labl8`

`labl1,,labl8` cross reference labels (max. 10 characters) for the parameters you may pass to this MIDAS procedure

**Note:** This must (!) be the first command in that procedure.

**Examples:** `CROSSREF FILEA FILEB METHOD`
If this is the first executable command line in a procedure, then the MIDAS commands:

`@@ FILTER GALAXY FILTGAL SMOOTH`
and

`@@ FILTER METHOD=SMOOTH FILEA=GALAXY FILEB=FILTGAL`
are equivalent and will both set parameters P1 to GALAXY, P2 to FILTGAL and P3 to SMOOTH.

`@@ FILTER P3=SMOOTH P1=GALAXY P2=FILTGAL`
then you obtain the same result as above, but no CROSSREF command is necessary inside the procedure

## DEFINE/LOCAL *core* 22-FEB-1993 KB

**Purpose:** Define and initialize a local keyword in a Midas procedure.

**Syntax:** `DEFINE/LOCAL key_def data [A] [lower_levels_flag]`

| | |
|---|---|
| `key_def` | keyname/keytype/first_elem/no_elem |
| `data` | data string |
| `A` | optional all_flag, if given all elements of the keyword are set to given data |
| `lower_levels_flag` | optional flag, if set to '+lower_levels', this local keyword is also defined at lower levels, i.e. in all procedures called from the one where the keyword is defined; by default local keywords are only known at the procedure level in which the procedure defining them is running |

**Note:** The names of local keywords have to be different from the names of the initial systems (global) keywords which are specified in the ASCII file 'syskeys.dat' and created when starting MIDAS. For all other keywords, local keywords take precedence over global keywords in case of name conflicts (i.e. same name).
When the procedure terminates, all keywords created via DEFINE/LOCAL in this procedure are deleted. This feature leads to problems if you want to look at the contents of local keywords, after the procedure has finished...
You have to execute the procedure in 'debug' mode (i.e. after having issued the command DEBUG/PROC) to be able to read the contents of local keywords.

**See also:** WRITE/KEYWORD, READ/KEYWORD, DEBUG/PROC

**Examples:** `DEFINE/LOCAL mykey/I/1/3 -2,-8,256`
Create an integer local keyword of 3 elements and initialize its elements to: -2, -8 and 256.

`DEFINE/LOCAL YOURKEY/D/1/30 1.d0 all`
Create a double precision local keyword of 30 elements and set all elements to 1.0 .

`DEFINE/LOCAL inherit/c/1/10 "I know you" ? +lower_levels`
Create a character local keyword of 10 elements (bytes) and make it known to all procedures called from the defining procedure.

## DEFINE/MAXPAR *core* 22-MAR-1993 KB

**Purpose:** Define the maximum no. of parameters for a procedure.

**Syntax:** `DEFINE/MAXPAR maxno`

| | |
|---|---|
| `maxno` | maximum no. of parameters, maxno in [1,8] |

**Note:** When the procedure is executed with more than 'maxno' parameters a warning message is displayed and the procedure continues.

**See also:** DEFINE/PARAMETER

**Examples:** `DEFINE/MAXPAR 3`
Tell MIDAS that the procedure should be executed with 3 parameters at most.

# DEFINE/PARAMETER

## DEFINE/PARAMETER *core* 24-JUL-1991 KB

**Purpose:** Define default, type and valid interval for parameter i.

**Syntax:** `DEFINE/PARAMETER Pi [default] [type_spec] [prompt_str] [lo_lim,hi_lim]`

| | |
|---|---|
| `Pi` | P1, P2, ..., or P8 |
| `default` | default value for Pi; defaulted to ? |
| `type_spec` | type/option of Pi<br>valid types are: I(mage), T(able), F(it), N(umber) or ?, if no type checking should be done; defaulted to ?<br>option = C or A;<br>C - check type of parameter Pi and continue, only update keyword PARSTAT(i)<br>A - check type of parameter Pi and abort if not equal to required type<br>Element i of the (system) keyword PARSTAT is set to 1 or 0, if parameter Pi is of the required type or not.<br>If type = ? the element PARSTAT(i) will always be set to 1.<br>If the option is A (for ABORT), which is the default, the procedure is aborted.<br>Besides type checking, also name translation is done, i.e. if Pi = #15 or &g , then Pi will obtain the relevant "real" frame name in execution of the command. |
| `prompt_str` | prompt string;<br>(has to be enclosed in " " if it contains blanks),<br>defaulted to ?;<br>if default = ?, i.e. no default defined, the user is prompted for the parameter using this prompt string |
| `lo_lim,hi_lim` | optional low, high limit of valid interval for parameter (only valid for single number parameters) |

**Note:** The type checking is only done on the first character of a parameter. So, if you have a numerical parameter, 4.3 as well as 4.a will pass the test, whereas a.4 will fail.

**Examples:** `DEFINE/PARAM P3 lola I`
Parameter P3 should contain an image name and is defaulted to lola. The procedure aborts, if not a valid image name

`DEFINE/PARAM P3 lola I/C`
Parameter P3 should contain an image name and is defaulted to lola. But the procedure continues also, if e.g. a number is entered. In that case PARSTAT(3) is set to 0 (may be tested in the procedure).

`DEFINE/PARAM P1 ? N "Enter number:" 1.3,22.9`
P1 should hold a number inside [1.3,22.9]. If not given, the user is prompted for this parameter.

DO *core* 21-JAN-1993 KB

**Purpose:** Define a DO loop (as in FORTRAN).

**Syntax:** `DO loopvar = start end [step]`

| | |
|---|---|
| `loopvar` | name of integer keyword serving as loop variable |
| `start` | starting integer value of loop, either integer constant or name of integer keyword |
| `end` | end integer value of loop, either integer constant or name of integer keyword |
| `step` | optional integer stepsize, either integer constant or name of integer keyword; defaulted to 1 |

**Note:** The DO block must be closed with an ENDDO command; please, note that contrary to FORTRAN the ENDDO must(!) be written as one word, i.e. END DO results in an error ...
Nested loops are supported up to 8 levels deep.
A DO loop is executed at least once, i.e. the test for loop termination is done in the end of the DO block.
A stepsize = 0 is legal and leads to an infinite loop unless the start value is already larger than the end value.

**Examples:**

```
DEFINE/LOCAL LL/I/1/1 0
DO LL = 12 22 2
WRITE/KEY INPUTI/I/LL/1 LL
ENDDO
```

This would fill keyword INPUTI(12,14,...,22) with the values 12,14,...,22.

```
DEFINE/LOCAL LL/I/1/1 0
OUTPUTI(4) = 10
DO LL = 22 OUTPUTI(4) -2
WRITE/OUT LL
ENDDO
```

This would display the numbers 22, 20, ..., 12, 10.

# ENTRY

## ENTRY *core* 18-JAN-1993 KB

**Purpose:** Define begin of MIDAS procedure in a file with different name than the procedure.

**Syntax:** `ENTRY proc`

`proc` name of procedure

**See also:** @@ procedure

**Note:** This procedure is then executed via '@@ file,proc'.

**Examples:** `ENTRY TUTTI`
in line 23 of the procedure 'test.prg'

`@@ test`
Will execute the commands stored in line 1 till line 22 of file 'test.prg' only, since the ENTRY command in line 23 has the same effect as a RETURN command.

`@@ test,tutti`
Will execute the commands stored in line 24 till last line of file 'test.prg' (if no other RETURN or ENTRY command is encountered).

GOTO *core* 15-JAN-1987 KB

**Purpose:** Branch to command line containing the "label" of the GOTO command.

**Syntax:** `GOTO label`

`label` a character string

**See also:** Chapter 3 of MIDAS Users Guide, volume A

**Note:** The label itself may be any string (max. 20 characters), the first character must be a letter and the last character must be the colon (:) - next command has to be on the next line.

**Examples:**
```
GOTO CHULO
...
CHULO:
LOAD/IMA mariposa
```
Jump to the line in the procedure containing the char. string "CHULO:" and execute the next command, i.e. the LOAD/IMA command.

# IF

| IF | *core* | 20-JAN-1993 KB |
|---|---|---|

**Purpose:** Execute conditional statement.

**Syntax:** `IF par1 op par2 command_string (a)`

```
IF par1 op par2 THEN (b)
   command_string
ELSEIF par1 op par2 THEN
   command_string
   ...
ELSE
   command_string
ENDIF
```

| | |
|---|---|
| `par1,par2` | name of a keyword (with parentheses) or a constant;<br>numerical constants begin with a digit;<br>character constants are enclosed in quotation marks (").<br>All character strings not enclosed in quotation marks are interpreted as keyword names !!! |
| `op` | logical operation, may be .EQ. or .NE. or .GE. or .GT. or .LE. or .LT. with the usual meaning (cf. FORTRAN) |
| `command_string` | valid MIDAS command line;<br>but remember that the total no. of 'tokens' per line is 10, therefore only "short" commands can be used for option (a) |

**Note:** Please, note that contrary to FORTRAN the ELSEIF and ENDIF must(!) be written as one word, i.e. ELSE IF results in an error ...
Logical operations can be combined with an .AND. or .OR. (but only a single .AND. or .OR. because we only have max. 10 tokens per command line), see the examples.

**See also:** Chapter 3 of the MIDAS Users Guide, Volume A

**Examples:** `IF INPUTR(6) .GT. 22.1 GOTO END`
will result in a branch to the command following the label END: if the element (6) of keyword INPUTR is greater than 22.1

```
IF INPUTR(6) .GT. 22.1 THEN
WRITE/KEY IN_B INPUTR(6)
WRITE/OUT IN_B
ELSEIF IN_A(3:4) .EQ. "AB" THEN
WRITE/OUT "here we are..."
ENDIF
```

`IF INPUTC(1:3) .EQ. "XXX" .OR. INPUTD(1) .LT. 0.0 RETURN`
we return from this procedure if either condition is true

```
IF INPUTI(10) .EQ. 10 .AND. INPUTI(11) .NE. 11 THEN
@@ proca
ELSE
@@ procb
ENDIF
```

---

## INQUIRE/KEYWORD *core* 05-MAY-1992 KB

**Purpose:** Get terminal input in a MIDAS procedure.

**Syntax:** `INQUIRE/KEYWORD key [prompt_string] [flush_opt]`

`key` complete keyword specification;
either name/type/first_elem/noval, e.g. INPUTI/I/1/2
or only keyword name, e.g. INPUTI, if the keyword exists already;
as many as 'noval' data values will be written into the keyword 'name' of given type, beginning at the first element (this is 1 if just keyword name is given).

`prompt_string` optional prompt string

`flush_opt` optional flag: if set to 'FLUSH' the input buffer is flushed, before prompting for the input, i.e. no type-ahead is possible

**See also:** WRITE/KEYWORD

**Note:** If the prompt string contains blanks, it must be enclosed in quotation marks.
The procedure does not continue until the input from the terminal has been entered (i.e. there is no timeout).
If you do not want to change the contents of the keyword, just hit RETURN. The keyword AUX_MODE(7) will contain the number of values entered.

**Examples:** `INQUIRE/KEY IN_A "Enter name of input image:"`
The procedure waits until user enters input and hits RETURN. Keyword in_a will contain that input.

`INQUIRE/KEY petrita/c/6/5 "Give me a name:" flush`
Will change the contents of petrita(6:10) with the user input; the input buffer is emptied before displaying the prompt string.

`INQUIRE/KEY inputr "Enter data values: "`
Assuming the user enters the following input terminated by RETURN:
16,2.3,-7.8
then the first 3 elements of keyword inputr are set to 16.0,2.3,-7.8;
AUX_MODE(7) will be set to 3.

# PAUSE

## PAUSE *core* 09-DEC-1994 KB

**Purpose:** Interrupt current Midas procedure and return to interactive level.

**Syntax:** `PAUSE`

**See also:** CONTINUE
Chapter 3 of MIDAS Users Guide, volume A

**Note:** After the procedure has been interrupted you can execute any other Midas command.
To continue again with the paused procedure, enter CONTINUE.
Only one procedure can be in a 'paused' state, i.e. after a procedure is stopped with a PAUSE command, you cannot execute another Midas procedure which also contains a PAUSE command.

**Examples:**
```
xyz...
PAUSE
abc...
```

The procedure will stop after execution of the command 'xyz...' and return to the interactive level.
The procedure will continue execution at line 'abc...' when entering the command CONTINUE.

RETURN *core* 14-JAN-1987 KB

**Purpose:** Return to calling procedure (or terminal) and pass up to 3 parameters back.

**Syntax:** `RETURN [par1] [par2] [par3]`

| | |
|---|---|
| `par1` | string which will be stored in the character keyword Q1 |
| `par2` | string which will be stored in the character keyword Q2 |
| `par3` | string which will be stored in the character keyword Q3 |

**See also:** Chapter 3 in the MIDAS Users Manual, Volume A

**Note:** This is a clean way of passing parameters back to the calling procedure and an alternative to using global keywords.
Use RETURN/EXIT to force control back to the terminal, i.e. stop procedure at any level

**Examples:** `Suppose we have two MIDAS procedures, proc1.prg and proc2.prg.`

Code of proc1.prg:
@@ proc2 !call proc2.prg
write/out Q1 = q1

Code of proc2.prg:
return moctezuma !return to calling procedure

Then executing proc1.prg via @@ proc1 would yield:
Q1 = moctezuma

# Appendix B

# Core Commands

## @@ *core* (@,@a,@s,@c) 16-MAY-1991 KB

**Purpose:** Execute a MIDAS command procedure.

**Syntax:** `@@ proc [par1] ... [par8]`

`proc` name of MIDAS command procedure, type defaulted to '.prg'
or 'proc,entry', if command "ENTRY entry" is used in proc.prg

`par1 ... par8` up to 8 actual parameters,
each parameter may be up to 80 chars. long, but total size of all parameters together is limited to 132 char.

**Note:** '@@ proc' searches for the procedure file in the current directory, if no path specification in file name,
if not found, MIDAS looks for the procedure file in MID_WORK;
'@ proc' searches in MID_PROC, which is the directory for the MIDAS system procedures;
'@a proc' searches in APP_PROC, which is the directory for the MIDAS application procedures;
'@s proc' searches in STD_PROC, which is the directory for the MIDAS standard reduction procedures;
'@c proc' searches in CON_PROC, which is the directory for the contributed procedures;
Don't forget that Unix is case sensitive, so procedure abc.prg is a different file than ABC.PRG on a Unix file system!

**See also:** RUN, HELP/CL, chapter 3 of Users Manual, volume A

**Examples:** `@ lobo AK 23.`
Execute procedure MID_PROC:lobo.prg with par1 = AK, par2 = 23.

`@a gallina mata hari`
Execute procedure APP_PROC:gallina.prg with par1 = mata and par2 = hari.

`@@ cabra.ALL`
Execute command procedure cabra.ALL or if no such file exists in the current directory, execute MID_WORK:cabra.ALL .

`@@ DRA2:[BIG3.OWN]perro 22.,33.`
Execute command procedure DRA2:[BIG3.OWN]perro.prg with par1 = 22.,33. on a VMS system.

`@@ /users/mine/specs/perro 22.,33.`
Execute command procedure /users/mine/specs/perro.prg with par1 = 22.,33. on a Unix system.

ADD/ACAT *core* 20-JUN-1991 KB

**Purpose:** Add entries to an ASCII file catalog.

**Syntax:** `ADD/ACAT [cat_name] frame_list`

| | |
|---|---|
| `cat_name` | name of ASCII file catalog;<br>defaulted to currently active ASCII file catalog |
| `frame_list` | list of frames to be added,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*) |

**Note:** The different options for the frame_list may not be mixed!

**See also:** SET/ACAT, CREATE/ACAT, CLEAR/ACAT, READ/ACAT, SUBTRACT/ACAT

**Examples:** `ADD/ACAT dec88 bar.chart`
Add an entry for file bar.chart to ASCII catalog dec88.cat

`ADD/ACAT dec88 zap*`
Add entries for all frames where the names begin with the string "zap" to catalog dec88.cat.

`ADD/ACAT dec88 nov89.cat`
Add entries for all frames which are in the ASCII catalog nov89.cat.

# ADD/FCAT

## ADD/FCAT *core* 25-SEP-1992 KB

**Purpose:** Add entries to a fitfile catalog.

**Syntax:** `ADD/FCAT [cat_name] file_list [lowstr,histr]`

| | |
|---|---|
| `cat_name` | name of Fit file Catalog, defaulted to currently active Fit file Catalog |
| `file_list` | list of files to be added,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.fit) |
| `lowstr,histr` | optional low and high strings;<br>if given, only fit files with names >= 'low' and <= 'high' will be added |

**Note:** The different options for the file_list may not be mixed!
Entries are added automatically to the currently "active" fit file catalog (SET/FCAT command makes a catalog active).

**See also:** SET/FCAT, CREATE/FCAT, CLEAR/FCAT, READ/FCAT, SUBTRACT/FCAT
ADD/ICAT, ADD/TCAT

**Examples:** `ADD/FCAT dec88 func01`
add an entry for fit file func01.fit to catalog dec88.cat.

`ADD/FCAT dec88 func0*`
Add entries for all fit files where the names begin with the string "func0" to catalog dec88.cat.

`ADD/FCAT dec88 d* demo0025.fit,demo0036.fit`
Add entries for fit files with names demo0025.fit -> demo0036.fit to catalog dec88.cat.

`ADD/FCAT dec88 nov89.cat`
Add entries for all files which are in the fit file catalog nov89.cat.

ADD/ICAT *core* 25-SEP-1992 KB

**Purpose:** Add entries to an image catalog.

**Syntax:** `ADD/ICAT [cat_name] frame_list [lowstr,histr]`

| | |
|---|---|
| `cat_name` | name of image catalog; defaulted to currently active image catalog |
| `frame_list` | list of frames to be added,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.bdf) |
| `lowstr,histr` | optional low and high strings;<br>if given, only frames with names >= 'low' and <= 'high' will be added |

**Note:** The different options for the frame_list may not be mixed!
Entries are added automatically to the currently "active" image catalog (SET/ICAT command makes a catalog active).

**See also:** CREATE/ICAT, SET/ICAT, CLEAR/ICAT, READ/ICAT, SUBTRACT/ICAT
ADD/TCAT, ADD/FCAT

**Examples:** `ADD/ICAT dec88 galax001,sun04,moon11.ima`
Add entries for image frames 'galax001.bdf', 'sun04.bdf' and 'moon11.ima' to catalog 'dec88.cat'.

`ADD/ICAT dec88 gal*`
Add entries for all image frames where the names begin with the string "gal" to catalog 'dec88.cat'.

`ADD/ICAT dec88 d* demo0025.bdf,demo0036.bdf`
Add entries for image frames with names 'demo0025.bdf'-> 'demo0036.bdf' to catalog 'dec88.cat'.

`ADD/ICAT dec88 nov89.cat`
Add entries for all frames which are in the image catalog 'nov89.cat' to image catalog 'dec88.cat'.

# ADD/TCAT

## ADD/TCAT *core* 25-SEP-1992 KB

**Purpose:** Add entries to a table catalog

**Syntax:** `ADD/TCAT [cat_name] table_list [lowstr,histr]`

| | |
|---|---|
| `cat_name` | name of table catalog; defaulted to currently active table catalog |
| `table_list` | list of tables to be added,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.tbl) |
| `lowstr,histr` | optional low and high strings;<br>if given, only tables with names >= 'low' and <= 'high' will be added |

**Note:** The different options for the frame_list may not be mixed!
Entries are added automatically to the currently "active" table catalog (SET/TCAT command makes a catalog active).

**See also:** SET/TCAT, CREATE/TCAT, CLEAR/TCAT, READ/TCAT, SUBTRACT/TCAT
ADD/ICAT, ADD/FCAT

**Examples:** `ADD/TCAT dec89 coords`
Add an entry for table file 'coords.tbl' to table catalog 'dec89.cat'.

`ADD/TCAT dec88 gal*`
Add entries for all tables where the names begin with the string "gal" to catalog 'dec88.cat'.

`ADD/TCAT dec88 d* demo0025.tbl,demo0036.tbl`
Add entries for tables with names 'demo0025.tbl' -> 'demo0036.tbl' to catalog 'dec88.cat'.

`ADD/TCAT dec88 nov89.cat`
Add entries for all tables which are in the table catalog 'nov89.cat' to table catalog 'dec88.cat'.

ALIGN/CENTER *core* 26-JAN-1990 KB

**Purpose:** Compute start coordinates for inframe so that center of inframe matches with center of refframe (full pixels).

**Subject:** Alignment

**Syntax:** `ALIGN/CENTER inframe refframe incent_x,_y refcent_x,_y`

| | |
|---|---|
| `inframe` | input frame |
| `refframe` | reference frame |
| `incent_x,_y` | center coordinates of inframe |
| `refcent_x,_y` | center coordinates of refframe |

**See also:** ALIGN/IMAGE, REBIN/ROTATE, CENTER/...

**Note:** The input frame and reference frame must have the same stepsize. The resulting start values are only displayed, descr. START of input frame is NOT updated.

**Examples:** `ALIGN/CENTER becerra vaca @10,@300 12.0,3245.0`
Calculate new start coordinates for frame 'becerra.bdf' so that the center pixels of 'becerra.bdf' and 'vaca.bdf' are aligned.

# ALIGN/IMAGE

## ALIGN/IMAGE *core* 05-MAY-1994 KB

**Purpose:** Compute transformation coefficients for rotation translation and scaling of an image.
The coefficients are computed using two tables with columns of world coordinates of common objects in the image to be aligned and the reference image, respectively.

**Subject:** Alignment, Transformation, Superposition

**Syntax:** `ALIGN/IMAGE intab reftab [option] [overlay_flag] [residual_flag]`

`intab` name of table and columns with world coordinates of object in the image to be aligned in the form: table[,:xlabel,:ylabel]
If the labels are omitted, they are defaulted to label :XCEN and :YCEN which are created by the Midas commands CENTER/GAUSS or CENTER/MOMENT

`reftab` name of table and columns with world coordinates of the same objects in the reference image in the form: table[,:xlabel,:ylabel].
If the labels are omitted, they are defaulted to :XCEN,:YCEN

`option` type of transformation to calculate as given below:
FREE, all parameters in the transformation are free;
EQUAL, the scaling factors in x and y are equal;
UNIT, the scaling factors in x and y are equal to 1;
SHIFT, the scaling factors in x and y are equal to 1 and rotation angle = 0.0;
defaulted to UNIT

`overlay_flag` OVER,[intensity] - plot positions of the object in table 'intab' transformed to the reference frame.
Points are plotted in the overlay plane with given intensity, default intensity = 255 (white); defaulted to no overlay
The reference frame must be displayed for this option to work!

`residual_flag` YES or NO - if you want to save the residuals calculated in ALIGN/IMAGE also to be stored in the input table 'intab' in columns labeled :XRESIDUAL, :YRESIDUAL and :RESIDUALS;
defaulted to NO

**See also:** Chapter 3 of Vol B of the MIDAS User's Guide,
CENTER/MOMENT, CENTER/GAUSS, REBIN/ROTATE, ALIGN/CENTER
TUTORIAL/ALIGN

**Note:** The character column IDENT is used to find matching pairs in the input tables. Therefore, this column must exist in both tables! For the SHIFT option one common object is sufficient while the other options require at least 3 common objects in the two tables.
The resulting rotation angle (in degrees), scaling factor in x and y are stored in the double precision keyword TRANSFRM(1,2,3), the transformation matrix in TRANSFRM(4-7), and the x,y translation in TRANSFRM(8-9).
The values in TRANSFRM may be used later on as input for the REBIN/ROTATE command.
Preparing the tables, applying ALIGN/IMAGE and finally rebinning the images accordingly is quite a complex process, therefore a tutorial exists (TUTORIAL/ALIGN) which shows the different steps involved when aligning two images.

**Examples:** `ALIGN/IMAGE c1 c2`

Use columns labeled :XCEN and :YCEN in tables 'c1.tbl' and 'c2.tbl' to obtain the world coordinates of common objects in the two images, say 'im1.bdf' and 'im2.bdf', from which tables 'c1.tbl' and 'c2.tbl' were built.
The method is defaulted to UNIT, so the scaling factors in x and y are forced to 1.0.
Then use: REBIN/ROTATE im1 im1r KEYWORD im2
to obtain image 'im1r.bdf' which is aligned with image 'im2.bdf'.

`ALIGN/IMAGE c1,:X_COORD,:Y_COORD c2,:X_COORD,:Y_COORD F`
Use columns labeled :X_COORD and :Y_COORD in tables 'c1.tbl' and 'c2.tbl' to obtain the world coordinates of common objects in the two images from which table 'c1.tbl' and 'c2.tbl' were built.
All transformation coefficients are free.

`ALIGN/IMAGE c1,:X_COORD,:Y_COORD c2,:X_COORD,:Y_COORD E OV`
As above but force equal scaling factors in x and y and display calculated positions (using the coefficients just obtained) for the reference points.

# APPLY/CONVERSION

## APPLY/CONVERSION *core* 08-JAN-1990 KB

**Purpose:** Convert a "mask" image to a table, or convert a table to a "mask" image.

**Subject:** Masks, tables defining regions of interest

**Syntax:** a) `APPLY/CONV IMTB ima tab threshold`

b) `APPLY/CONV TBIM tab ima npx1,npx2 sta1,sta2,stp1,stp2 bg,fg`

`fromto_flag` IMTB for image to table (a), or TBIM for table to image (b); defaulted to IMTB

`(a) ima` input frame defining a "mask" where regions of interest are specified by a pixel value above the threshold

`(a) tab` output table, columns :XSTART, :YSTART, :XEND and :YEND hold the world coords. of the regions of interest

`(a) threshold` threshold value for regions of interest; defaulted to 1.0

`(b) tab` input table, columns :XSTART, :YSTART, :XEND and :YEND hold the world coords. of the regions of interest

`(b) ima` output frame defining a "mask" where regions of interest have foreground pixel value, the rest has background pixel value

`(b) npx1,npx2` no. of pixels in x,y dimension for output image; defaulted to 200,200

`(b) sta1,sta2,stp1,stp2` start and step values in x,y; defaulted to 0.0,0.0,1.0,1.0

`(b) bg,fg` background, foreground values to define regions; defaulted to 0.0,1.0

**Note:** Currently the regions of interest cannot overlap if you convert from a "mask" image to a table. The main use of these commands is to prepare a table for all the commands which accept regions of interest specified in a table, like e.g. MODIFY/PIXEL.

**See also:** MODIFY, STATISTICS commands

**Examples:** `APPLY/CONV IMTB masky roi 63.45`
Store the start, end (world) coordinates of all regions in the frame 'masky.bdf' with a pixel value > or = 63.45 in the columns labeled :XSTART, :YSTART, :XEND, :YEND of the new table 'roi.tbl'.

`APPLY/CONV TBIM coords quetzal 400,400 ? 16.6,33.3`
Create the 400*400 frame 'quetzal.bdf' (start 0.0,0.0 and step sizes 1.0,1.0). All regions specified in table 'coords.tbl' get pixel value 33.3, the rest of the image is set to 16.6 .

## APPLY/EDGE *core* 20-JUN-1991 KB

**Purpose:** Do edge detection on an image.

**Subject:** Edge detection, filter

**Syntax:** `APPLY/EDGE inframe outframe [thresh]`

| | |
|---|---|
| `inframe` | input frame |
| `outframe` | output frame |
| `thresh` | threshold value for edge detection;<br>default = (max-min)/2 of input frame |

**Note:** Apply simple thresholding algorithm (see the book by Gonzalez and Wintz, Digital Image Processing, Chapter 7) to detect edges in an image.

**See also:** FILTER/... commands

**Examples:** `APPLY/EDGE manzana durazno`
Do edge detection on image manzana.bdf and store result in durazno.bdf, use 0.5*(max-min) of pixels in manzana.bdf as threshold.

# APPLY/MAP

## APPLY/MAP *core* 12-NOV-1987 KB

**Purpose:** use an image frame like a Lookup Table

**Syntax:** `APPLY/MAP outframe = inframe mapframe control_flags`

`outframe` output frame

`inframe` input frame

`mapframe` 1-dim frame which serves as a pseudo LookupTable

`control_flags` two variables controlling
(1) mapping of input pixels with intensities outside the coord. space of the map frame,
0: map these pixels to start and end of map frame
1: leave these pixels unchanged
(2) indicating equidistant or other coord.space of map frame
0: 1-dim map frame, coord.space defined by descriptor START and STEP
1: 2-dim map frame, 1. line holds map coords, 2. line holds map intensities
2: 2-dim map frame, 1. line holds valid intervals, 2. line holds map intensities for endpoints of intervals (in between interpolation);
defaulted to 0,0

**See also:** COMPUTE/IMAGE, REPLACE/IMAGE, LOAD/LUT

**Note:** The intensity of each pixel of the input frame is interpreted as a world coordinate, 'wx' in the coordinate space of the mapframe.
The corresponding output pixel will either have the intensity of the pixel in the mapframe with coords closest to 'wx' or will be interpolated from the intensities of the enclosing coords interval.

**Examples:** `APPLY/MAP new = old lut`
Map frame 'old.bdf' via 1-dim frame 'lut.bdf' to obtain result frame 'new.bdf'. Descriptors START + STEP of 'lut.bdf' determine mapping index.

`APPLY/MAP new = old lut 1,0`
As above, but all pixels with an intensity < start_map or > end_map are left unmapped (i.e. unchanged).

`APPLY/MAP new = old lut 0,1`
Determine corresponding map pixel from map coordinate which is closest to input pixel intensity. 'lut.bdf' is a 2-dim frame, first line holds mapping coords. and second line mapping intensities.

`APPLY/MAP new = old lut 0,2`
Find map coordinate interval which contains input pixel intensity, and interpolate output pixel from corresponding map intensities. 'lut.bdf' is a 2-dim frame, first line holds mapping coords. intervals (there may be holes where nothing is changed) and second line holds mapping intensities for endpoints of the intervals.

## APPLY/THIN *core* 05-MAY-1989 KB

**Purpose:** Apply thinning algorithm to input frame.

**Syntax:** `APPLY/THIN inframe outframe`

| | |
|---|---|
| `inframe` | image with only 0's and 1's (or 0.0 and 1.0's) |
| `outframe` | result containing the thinned input data |

**See also:** APPLY/EDGE

**Note:** Algorithm from T. Pavlidis

**Examples:** `APPLY/THIN aguila zopilote`
Apply thinning algorithm to data of frame 'aguila.bdf' and store results into new frame 'zopilote.bdf' .

## ASSIGN/DEFAULT *core* 04-JAN-1991 KB

**Purpose:** Assign default devices.

**Syntax:** `ASSIGN/DEFAULT`

**See also:** ASSIGN/PRINT, ASSIGN/GRAPHICS, ASSIGN/IN, ASSIGN/DISPLAY

**Note:** This command uses all the commands above with the default values.

**Examples:** `ASSIGN/DEFAULT`
Assign the line printer as print device, image display as output for image display, etc.

# ASSIGN/DISPLAY

## ASSIGN/DISPLAY *core* 04-JAN-1991 KB

**Purpose:** Define output device for displaying images.

**Syntax:** `ASSIGN/DISPLAY [dev] [file_name]`

| | |
|---|---|
| `dev` | display device (may be shortened to the first 3 characters):<br>DISPLAY or D,display_id in X11 environment, display_id in [0,9];<br>VERSATEC (only supported under VMS);<br>LASER for grayscale Postscript laser;<br>COLOUR for colour Postscript laser;<br>SLIDE for colour film recorder;<br>FILE, if output should go to file indicated in next parameter;<br>defaulted to DISPLAY |
| `file_name` | name of file where to store the image in, only if parameter 'dev' is set to FILE |

**See also:** ASSIGN/PRINT, ASSIGN/DEFAULT, LOAD/IMAGE, COPY/DISPLAY

**Note:** The display assignment will be used in subsequent LOAD/IMAGE and LOAD/ITT commands.
Instead of the generic device names you can also use the system name of a hardcopy device at your site directly.
For example, at ESO "ps2usr1" would be possible.
The actual devices accessed via the generic names LASER, COLOUR and SLIDE depend upon how MIDAS was installed at your site.
Use the command "HELP [Printers]" to get a detailed list of all actually available hardcopy devices at your site.

**Examples:** `ASSIGN/DISP colour`
Assign the "default" colour laser printer as image output device.

`ASSIGN/DISP D,2`
Assign display window 2 as image output device (assuming a X11 workstation as image display).

`ASSIGN/DISP FILE display`
Subsequent LOAD/IMAGE command will fill the file 'display.bdf'.

`ASSIGN/DISP ps2usr1`
Assuming that you have a grayscale Postscript printer at your site with the name "ps2usr1", this defines ps2usr1 as output device.

## ASSIGN/GRAPHICS *core* 15-JUL-1988 RHW

**Purpose:** Define the graphic device to which all subsequent plots will be sent.

**Subject:** Hard copy, graphics, HELP [PRINTERS]

**Syntax:** `ASSIGN/GRAPHICS [device] [option]`

| | |
|---|---|
| `device` | plot device; Possible choices can be:<br>T[ERMINAL] graphics terminal emulation. D[ISPLAY] the Deanza display or, for workstations, image window 0;<br>G[WINDOW,n] graphics window n (default 0);<br>D[WINDOW,n] image window n (default 0);<br>one of the HARDCOPY devices (see below);<br>POSTSCRIPT generic postscript device; NULL the null device; in this case a plot file will be created. Handy if you have no graphic display capabilities available. |
| `option` | spool option; Possible choices are: SPOOL plots routed immediately (default); NOSPOOL plots will be kept on disk and only routed after a COPY/GRAPHICS command is given. |

**Note:** The choice between SPOOL and NOSPOOL can only been made if a hardcopy device has been assigned. The NOSPOOL parameter can be useful in cases where the user wants to produce a plot with a number of overplots on a hardcopy device. In that case the plot files are not routed to the device but stored on disk. After finishing the user can make the hardcopy with the COPY/GRAPHICS command.
Hardcopy devices should be specified by the system device names. For a number of devices default (MIDAS) names have been implemented. A complete overview of the available devices and their names can be obtained with the command HELP [PRINTERS].
All postscript printers and the device POSTSCRIPT offer the possibility to print in portrait or in landscape mode. In order to get the desired format one has to extend the printer name with ".l" for landscape or ".p" for portrait mode. Default (no extension given) is landscape mode.
The scales of a plot may change if a plot is sent to a plot device other than the original one (pre-specified by ASSIGN/GRAPHICS). In general the axis ratio of the frame will have changed, and hence a square frame WILL NOT BE A SQUARE FRAME ANYMORE if you use COPY/GRAPHICS to dump your plot on another device.

**See also:** COPY/GRAPHICS, SET/GRAPHICS, SHOW/GRAPHICS, HELP [PRINTERS]

**Examples:** `ASSIGN/GRAPHICS G,1`
plots are produced on MIDAS graphics window 1. The window will be created if it is not available yet.

`ASSIGN/GRAPHICS ps2usr1`
plots (in landscape mode) will be spooled to the printer/plotter ps2usr1; this is a system device name and may not be available for this MIDAS installation; check with HELP [PRINTERS] for the device names.

`ASSIGN/GRAPHICS ps2usr1.p`
The same as for the first example. However with the extension ".p" plots will be printed in portrait mode.

`ASSIGN/GRAPHICS LASER`
plots will be spooled to the default printer/plotter (LASER); check with HELP [PRINTERS] which device that actually is. Also here extensions ".p" or ".l" are allowed.

`ASSIGN/GRAPHICS POSTSCRIPT`

# ASSIGN/INPUT

Plots will be stored in the encapsulated postscript file postscript.ps. They can be included in a LateX document or be sent to any postcript device. Also here extensions ".p" or ".l" are allowed.

`ASSIGN/GRAP VERSA NOSPOOL`
plots are intended for the Versatec but will not be spooled. The plot files will be kept on disk and can be spooled by the command COPY/GRAPH device_name.

---

ASSIGN/INPUT *core* 04-JAN-1991 KB

**Purpose:** Define input device from where data is to be taken when writing to a MIDAS data structure.

**Syntax:** `ASSIGN/INPUT [dev] [file_name]`

| | |
|---|---|
| `dev` | input device (may be shortened to the first 3 characters):<br>TERMINAL, data input from terminal;<br>FILE, if input from file indicated in next parameter; |
| `file_name` | name of ASCII file to get input from |

**See also:** ASSIGN/PRINT, WRITE/IMAGE

**Note:** The input assignment will be used in subsequent WRITE/IMAGE commands. the data input will be from a file, the data has to be in free format separated by a blank or comma.
There is also the possibility to get input from an ASCII file instead from the terminal via the syntax: Midas_commmand_line <file
Use HELP [Host+Midas] for more info about I/O redirection in Midas.

**Examples:** `ASSIGN/INPUT FILE raton.dat`
The next WRITE/IMAGE command will use the contents of the ASCII file 'raton.dat'.

ASSIGN/PRINT *core* 9-JUL-1991 KB

**Purpose:** Define output device for printing.

**Syntax:** `ASSIGN/PRINT [dev [file_name]]`

| | |
|---|---|
| `dev` | print device (may be shortened to the first 3 characters):<br>LPRINT, the default line printer;<br>TERMINAL, user terminal;<br>LASER, the default laser printer;<br>FILE, if output should go to file indicated in next parameter;<br>defaulted to LPRINT |
| `file_name` | name of ASCII file to store data in for dev = FILE |

**See also:** ASSIGN/DEFAULT, PRINT/LOG, PRINT/TABLE, PRINT/IMAGE, PRINT/KEYWORD, PRINT/DESCR, PRINT/ICAT, PRINT/HELP

**Note:** This assignment will be used in all subsequent PRINT commands.
Instead of the generic device names you can also use the system name of a printer at your site directly.
For example, at ESO "ps2usr1" would be possible.
Use the command "HELP [Printers]" to get a detailed list of all actually available printers at your site.

**Examples:** `ASSIGN/PRINT`
Send all output from a subsequent PRINT command, e.g. PRINT/IMA to the the line printer.

`ASSIGN/PRINT FILE raton.dat`
Send all output from subsequent PRINT commands to the ASCII file 'raton.dat', this file will be created by MIDAS.

# AVERAGE/AVERAGE

## AVERAGE/AVERAGE *core* 19-JUL-1990 KB

**Purpose:** Compute simple average of all pixels in a subimage.

**Syntax:** `AVERAGE/AVERAGE [in_specs] [out_specs] [out_opt] [draw_flag]`

| | |
|---|---|
| `in_specs` | input specifications,<br>(a) CURSOR, if subimages are chosen via the cursor rectangle,<br>(b) image,table if subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND,<br>defaulted to CURSOR |
| `out_specs` | output specifications,<br>(c) :label indicating that the result value should be stored in column :label of the table given in in_specs (b)<br>(d) table,:label for cursor input (a) only,<br>(e) descriptor if the results should be stored in a real descr. of the involved image, data will be stored as xstart, ystart, xend, yend and average value of subimage<br>(f) if omitted, to just display values on the terminal |
| `out_opt` | A, append data to a descriptor, else start at the beginning, for (e) only<br>= C, store also x-,y-center into columns labeled :XCEN and :YCEN of given table, for (c) + (d) |
| `draw_flag` | Y(es) or N(o) for drawing or not drawing the subimages in the overlay plane |

**See also:** AVERAGE/KAPPA, AVERAGE/MEDIAN, AVERAGE/WEIGHTS, AVERAGE/IMAGE, AVERAGE/COLUMN, AVERAGE/ROW

**Note:** Currently,this command is not implemented for 1-dim frames.
When using the cursor input option, a cursor rectangle will appear on the screen.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start calculation on subimage. Set both cursors on to move cursor window. Change cursor size by setting one cursor off, then move joystick. To exit set both cursors off and press ENTER .
For X-Windows: Press ENTER button on the mouse to start calculation on subimage. Use the mouse to move cursor window. Use the arrow keys to change the size of the cursor window. To exit press the EXIT button on the mouse.

**Examples:** `(1) AVERAGE/AVE`
Use cursor to define subimages and display start- and end-coord of subimage and average value on terminal only.

`(2) AVERAGE/AVE CURSOR VALUES`
As (1) but write data to the beginning of real descriptor VALUES of displayed image.

`(3) AVERAGE/AVE CURSOR values A N`
As (2) but append data to the end of real descriptor values and do not draw the borders of the subimages into the overlay plane.

`(4) AVERAGE/AVE CURSOR values,:MEAN`
As (1) but store data into table values.tbl (this will be created by MIDAS), result value will be stored into column labeled :MEAN.

`(5) AVERAGE/AVE CURSOR values,:MEAN C`
As (4) but store also center coordinates into columns labeled :XCEN and :YCEN.

`(6) AVERAGE/AVE ccd001,flats`
Use columns :XSTART, :YSTART, :XEND and :YEND of table flats.tbl to define the subimages of frame ccd001.bdf, just display results.

(7) `AVERAGE/AVE ccd001,flats values`
As (2) but input from table as in (6).

(8) `AVERAGE/AVE ccd001,flats :MEAN`
As (7) but store result into column :MEAN of table flats.tbl.

---

## AVERAGE/COLUMN *core* 17-APR-1991 JDP

**Purpose:** Average columns in a selected range of the input image. The output is optionally the sum of the columns.

**Syntax:** `AVERAGE/COLUMN out = in [start,end] [SUM]`

| | |
|---|---|
| `out` | resulting 1 dimensional frame |
| `in` | image to be averaged |
| `start` | first column, defaulted to the first column of the frame |
| `end` | last column, defaulted to the last column of the frame |
| `SUM` | optional parameter to define the output as sum of the columns. By default the output is the average of them. |

**See also:** AVERAGE/ROW

**Note:** First and last columns can be specified in pixels or world coords.

**Examples:** `AVERAGE/COLUMN out = in @1,@10`
Average the columns of frame in.bdf, store result in out.bdf

# AVERAGE/IMAGES

## AVERAGE/IMAGES *core* 26-MAR-1992 KB

**Purpose:** Calculate the average of images - size of result frame is either intersection or union of all image sizes.

**Syntax:** `AVERAGE/IMAGES out = in_specs [merge] [null] [av_option] [dat_intval]`

| | |
|---|---|
| `out` | result frame |
| `in_specs` | specification of input frames, either<br>in1,in2,...inj or<br>catalog.cat = name of catalog containing the input frames; |
| `merge` | M (for Merge), size of result frame will be set to the union of all input frames, else the size will be the intersection of all input frames;<br>defaulted to N (No merge) |
| `null` | value for undefined pixels, also the real keyword NULL(2) will be updated accordingly;<br>if set to '+', undefined pixels are set to the value of the pixel calculated before;<br>defaulted to the actual contents of keyword NULL(2) |
| `av_option` | av_spec,av_low,av_hi,av_switch;<br>av_spec - Average (a), Minimum (b), Maximum (c) or Median (d),<br>(a) take average of of all valid pixels;<br>(b) take minimum of of all valid pixels;<br>(c) take maximum of of all valid pixels;<br>for (b) + (c) av_low may be set to indicate that the average (instead of the min/max) should be taken of<br>minimum/maximum and 'av_low' next higher/lower pixels<br>(d) take median of all valid pixels;<br>for (d) av_low, av_hi may be set to indicate that the average (instead of the median) should be taken of all the pixels in an interval around the median pixel, to indicate if the interval refers to the indices (order) of the pixels or their values, av_switch is set to INDEX or DATA, if 'av_switch' not given, it is used as INDEX;<br>except for (d) with av_switch = DATA, 'av_low' and 'av_hi' must be non-negative values;<br>defaulted to AV |
| `dat_intval` | rlow,rhigh - optional interval for input pixels;<br>if omitted (the default) all pixels in the input frames are used for the calculations |

**See also:** AVERAGE/WEIGHTS, AVERAGE/ROW, AVERAGE/WINDOW

**Note:** Undefined pixels result from non-overlapping areas when using the Merge option or from the constraints specified in the average options and data interval.
Overlapping areas will be determined via the world coordinates of the frames, i.e. via descriptors START and STEP.
The input frames are all opened simultaneously to speed up the averaging process. Thus, the no. of frames we can work on depends on a system variable (FOPEN_MAX on Unix systems, stored in stdio.h). In 'averag.exe' a max. of 64 frames is foreseen which is the value of FOPEN_MAX on many Unix systems. But since there are also other files open already (e.g. keyword file, logfile) the practical limit is about 54 files. However, if like in Solaris the max. no. of opened files is set to 20, only 12 or 13 frames can be opened simultaneously. Therefore, 54 is only the upper limit.
If the keyword MID$SPEC contains the string 'DEBUG', a MIDAS image named 'averdumy.dum' will be created containing the valid pixel count for each x,y position of the result frame.
If av_option=median, the index of the median pixel is calculated as (nopix+1)/2 with nopix = no. of pixels used.

**Examples:** `AVERAGE/IMAGES av1 = ccd01,ccd02,ccdrbv,flata,flatb,flatc,flatd`
Compute average of given input frames on their overlapping area. Size of result frame 'av1.bdf' will be the overlapping part of all input frames.
If the above input frames had entries in the image catalog 'ccdflat.cat', and only these, "AVERAGE/IMAGES av1 = ccdflat.cat"
would be an equivalent command ...

`AVERAGE/IMAGES av1 = ccdflat.cat ?  -99.9 min 0.3,7.23`
Take minimum of input frames, but use only pixels in the interval [0.3,7.23] for finding the minimum. If for a pixel all corresponding input pixels are outside the given interval, the pixel is set to -99.9.

`AVERAGE/IMAGES av1 = ccdflat.cat ?  -99.9 min,2 0.3,7.23`
As above, but resulting pixel will be the average of the 3 lowest pixels. Less than 3 pixels may be used for the averaging depending upon how many valid pixels exist at a given x,y position of the result frame.
For example, if the first pixels in the 7 input frames had the values 1.0, 4.4, 2.1, 3.0, 2.0, 4.0, 5.6 then the first pixel in the result frame would be calculated as (1.0 + 2.0 + 2.1)/3 .

`AVERAGE/IMAGES av1 = ccdflat.cat ?  -99.9 max,4 0.3,7.23`
As above, but resulting pixel will be the average of the 5 highest pixels.

`AVERAGE/IMAGES av1 = ccdflat.cat ?  + max,4 0.3,7.23`
As above, but undefined pixels are set to the value of the pixel calculated before. If the very first result pixel is an undefined pixel, it is set to 0.0 .

`AVERAGE/IMAGES av1 = ccdflat.cat ?  + median,1,2 0.3,7.23`
As above, but resulting pixel will be the average of (if possible) the 4 pixels in the ordered interval [median-1,median+2]. Using the numbers from the example above, the first result pixel would be (2.1 + 3.0 + 4.0 + 4.4)/4 = 3.375 (median is 3.0).

`AVERAGE/IMAGES av1 = ccdflat.cat ?  + median,1,2,data 0.3,7.23`
As above, but resulting pixel will be the average of all pixels within the interval [median_val-1.0,median_val+2.0]. Using the same numbers as above, the first result pixel would be
(2.0 + 2.1 + 3.0 + 4.0 + 4.4)/5 = 3.10

`AVERAGE/IMAGES av2 = ccdflat.cat M 0.33`
Compute average of given input frames on their combined area. Size of result frame 'av2.bdf' will be the union of all frames with entries in image catalog 'ccdflat.cat'. Undefined pixels are set to 0.33
Here only pixels at x,y-positions with no corresponding input pixels are undefined, since parameters 'av_option' and 'dat_intval' are not given.

# AVERAGE/KAPPA

## AVERAGE/KAPPA *core* 19-JUL-1990 KB

**Purpose:** Compute average of all pixels in a subimage via kappa-sigma algorithm.

**Syntax:** `AVERAGE/KAPPA [in_specs] [out_specs] [out_opt] [draw_flag] [no_iter]`

| | |
|---|---|
| `in_specs` | input specifications,<br>(a) CURSOR, if subimages are chosen via the cursor rectangle,<br>(b) image,table if subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND,<br>defaulted to CURSOR |
| `out_specs` | output specifications,<br>(c) :label indicating that the result value should be stored in column :label of the table given in in_specs (b)<br>(d) table,:label for cursor input (a) only,<br>(e) descriptor if the results should be stored in a real descr. of the involved image, data will be stored as xstart, ystart, xend, yend and average value of subimage<br>(f) if omitted, to just display values on the terminal |
| `out_opt` | A, append data to a descriptor, else start at the beginning, for (e) only<br>= C, store also x-,y-center into columns labeled :XCEN and :YCEN of given table, for (c) + (d) |
| `draw_flag` | Y(es) or N(o) for drawing or not drawing the subimages in the overlay plane |
| `no_iter` | no. of iterations for 2*sigma clipping, defaulted to 1 |

**See also:** AVERAGE/AVERAGE, AVERAGE/MEDIAN, AVERAGE/WEIGHTS, AVERAGE/IMAGE, AVERAGE/COLUMN, AVERAGE/ROW

**Note:** Currently,this command is not implemented for 1-dim frames.
When using the cursor input option, a cursor rectangle will appear on the screen.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start calculation on subimage. Set both cursors on to move cursor window. Change cursor size by setting one cursor off, then move joystick. To exit set both cursors off and press ENTER .
For X-Windows: Press ENTER button on the mouse to start calculation on subimage. Use the mouse to move cursor window. Use the arrow keys to change the size of the cursor window. To exit press the EXIT button on the mouse.

**Examples:** `(1) AVERAGE/KAPPA`
Use cursor to define subimages and display start- and end-coords of subimage and average value on terminal only.

`(2) AVERAGE/KAPPA CURSOR VALUES`
As (1) but write data to the beginning of real descriptor VALUES of displayed image.

`(3) AVERAGE/KAPPA CURSOR values A`
As (2) but append data to the end of real descriptor values.

`(4) AVERAGE/KAPPA CURSOR values,:MEAN`
As (1) but store data into table 'values.tbl' (will be created by MIDAS), result value will be stored into column labeled :MEAN.

`(5) AVERAGE/KAPPA CURSOR values,:MEAN C`
As (4) but store also center coordinates into columns labeled :XCEN and :YCEN.

`(6) AVERAGE/KAPPA ccd001,flats`

Use columns :XSTART, :YSTART, :XEND and :YEND of table 'flats.tbl' to define the subimages of frame 'ccd001.bdf', just display results.

`(7) AVERAGE/KAPPA ccd001,flats values`
As (2) but input from table as in (6).

`(8) AVERAGE/KAPPA ccd001,flats :MEAN`
As (7) but store result into column :MEAN of table 'flats.tbl'.

# AVERAGE/MEDIAN

## AVERAGE/MEDIAN *core* 19-JUL-1990 KB

**Purpose:** Compute average of all pixels in a subimage by calculating the median value.

**Syntax:** `AVERAGE/MEDIAN [in_specs] [out_specs] [out_opt] [draw_flag]`

| | |
|---|---|
| `in_specs` | input specifications,<br>(a) CURSOR, if subimages are chosen via the cursor rectangle,<br>(b) image,table if subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND,<br>defaulted to CURSOR |
| `out_specs` | output specifications,<br>(c) :label indicating that the result value should be stored in column :label of the table given in in_specs (b)<br>(d) table,:label for cursor input (a) only,<br>(e) descriptor if the results should be stored in a real descr. of the involved image, data will be stored as xstart, ystart, xend, yend and average value of subimage<br>(f) if omitted, to just display values on the terminal |
| `out_opt` | A, append data to a descriptor, else start at the beginning, for (e) only<br>= C, store also x-,y-center into columns labeled :XCEN and :YCEN of given table, for (c) + (d) |
| `draw_flag` | Y(es) or N(o) for drawing or not drawing the subimages in the overlay plane |

**See also:** AVERAGE/KAPPA, AVERAGE/AVERAGE, AVERAGE/WEIGHTS, AVERAGE/IMAGE, AVERAGE/COLUMN, AVERAGE/ROW

**Note:** Currently,this command is not implemented for 1-dim frames.
When using the cursor input option, a cursor rectangle will appear on the screen.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start calculation on subimage. Set both cursors on to move cursor window. Change cursor size by setting one cursor off, then move joystick. To exit set both cursors off and press ENTER .
For X-Windows: Press ENTER button on the mouse to start calculation on subimage. Use the mouse to move cursor window. Use the arrow keys to change the size of the cursor window. To exit press the EXIT button on the mouse.

**Examples:** `(1) AVERAGE/MEDIAN`
Use cursor to define subimages and display start- and end-coords of subimage and average value on terminal only.

`(2) AVERAGE/MEDIAN CURSOR VALUES`
As (1) but write data to the beginning of real descriptor VALUES of displayed image.

`(3) AVERAGE/MEDIAN CURSOR values A`
As (2) but append data to the end of real descriptor values.

`(4) AVERAGE/MEDIAN CURSOR values,:MEAN`
As (1) but store data into table 'values.tbl' (will be created by MIDAS), result value will be stored into column labeled :MEAN.

`(5) AVERAGE/MEDIAN CURSOR values,:MEAN C`
As (4) but store also center coordinates into columns labeled :XCEN and :YCEN.

`(6) AVERAGE/MEDIAN ccd001,flats`
Use columns :XSTART, :YSTART, :XEND and :YEND of table 'flats.tbl' to define the subimages of frame 'ccd001.bdf', just display results.

(7) `AVERAGE/MEDIAN ccd001,flats values`
As (2) but input from table as in (6).

(8) `AVERAGE/MEDIAN ccd001,flats :MEAN`
As (7) but store result into column :MEAN of table 'flats.tbl'.

---

## AVERAGE/ROW *core* 17-APR-1991 JDP

**Purpose:** Average rows in a selected range of the input image. The output is optionally the sum of the rows.

**Syntax:** `AVERAGE/ROW out = in [start,end] [SUM]`

| | |
|---|---|
| `out` | resulting 1 dimensional frame |
| `in` | image to be averaged |
| `start` | first row, defaulted to the first row of the frame |
| `end` | last row, defaulted to the last row of the frame |
| `SUM` | optional parameter to define the output as sum of the rows. By default the output is the average of them. |

**See also:** AVERAGE/COLUMN

**Note:** First and last row can be specified in pixels or world coords.

**Examples:** `AVERAGE/ROW out = in 1.,10.`
Average the rows of frame in.bdf, store result in out.bdf

# AVERAGE/WEIGHTS

## AVERAGE/WEIGHTS *core* 26-MAR-1992 KB

**Purpose:** Calculate the weighted average of images - size of result frame is either the intersection or the union of all image sizes.

**Syntax:** `AVERAGE/WEIGHTS out = in_specs [merge] [null] [av_option] [dat_intval]`

| | |
|---|---|
| `out` | result frame |
| `in_specs` | specification of input frames, either<br>in1,in2,...inj or<br>catalog.cat = name of catalog containing the input frames; |
| `merge` | M (for Merge), size of result frame will be set to the union of all input frames, else the size will be the intersection of all input frames;<br>defaulted to N (No merge) |
| `null` | value for undefined pixels, also the real keyword NULL(2) will be updated accordingly;<br>if set to '+', undefined pixels are set to the value of the pixel calculated before;<br>defaulted to the actual contents of keyword NULL(2) |
| `av_option` | av_spec,av_low,av_hi,av_switch;<br>av_spec - Average (a), Minimum (b), Maximum (c) or Median (d),<br>(a) take average of of all valid pixels;<br>(b) take minimum of of all valid pixels;<br>(c) take maximum of of all valid pixels;<br>for (b) + (c) av_low may be set to indicate that the average (instead of the min/max) should be taken of<br>minimum/maximum and 'av_low' next higher/lower pixels<br>(d) take median of all valid pixels;<br>for (d) av_low, av_hi may be set to indicate that the average (instead of the median) should be taken of all the pixels in an interval around the median pixel,<br>to indicate if the interval refers to the indices (order) of the pixels or their values, av_switch is set to INDEX or DATA, if 'av_switch' not given, it is used as INDEX;<br>except for (d) with av_switch = DATA, 'av_low' and 'av_hi' must be non-negative values;<br>defaulted to AV |
| `dat_intval` | rlow,rhigh - optional interval for input pixels;<br>if omitted (the default) all pixels in the input frames are used for the calculations |

**See also:** AVERAGE/IMAGES, COMPUTE/WEIGHTS

**Note:** The weights of the input frames are taken from real descriptor WEIGHT (of 1 element) of each frame, this descriptor may either be filled via the command WRITE/DESCR for each frame individually or via the command COMPUTE/WEIGHTS for a group of frames.
Except for the weighting factors, the calculations are exactly the same as for AVERAGE/IMAGES.
Undefined pixels result from non-overlapping areas when using the Merge option or from the constraints specified in the average options and data interval.
Overlapping areas will be determined via the world coordinates of the frames, i.e. via descriptors START and STEP.
The input frames are all opened simultaneously to speed up the averaging process. Thus, the no. of frames we can work on depends on a system variable (FOPEN_MAX on Unix systems, stored in stdio.h). In 'averag.exe' a max. of 64 frames is foreseen which is the value of FOPEN_MAX on many Unix systems. But since there are also other files open already (e.g. keyword file, logfile) the practical limit is about 54 files. However, if like in Solaris the max. no. of opened files is set to 20, only 12 or 13 frames can be opened simultaneously. Therefore, 54 is only the upper limit.
If the keyword MID$SPEC contains the string 'DEBUG', a MIDAS image named 'averdumy.dum' will be created containing the valid pixel count for each x,y position of the result frame.
If av_option=median, the index of the median pixel is calculated as (nopix+1)/2 with nopix = no. of pixels used.

**Examples:** `AVERAGE/WEIGHTS av1 = ccd01,ccd02,ccdrbv,flata,flatb,flatc,flatd`
Compute weighted average of given input frames on their overlapping area. Size of result frame 'av1.bdf' will be the overlapping part of all input frames.
If the above input frames had entries in the image catalog 'ccdflat.cat', and only these, "AVERAGE/IMAGES av1 = ccdflat.cat"
would be an equivalent command ...

`AVERAGE/WEIGHTS av1 = ccdflat.cat ?  -99.9 min 0.3,7.23`
Take minimum of input frames, but use only pixels in the interval [0.3,7.23] for finding the minimum. If for a pixel all corresponding input pixels are outside the given interval, the pixel is set to -99.9.

`AVERAGE/WEIGHTS av1 = ccdflat.cat ?  -99.9 min,2 0.3,7.23`
As above, but resulting pixel will be the average of the 3 lowest pixels. Less than 3 pixels may be used for the averaging depending upon how many valid pixels exist at a given x,y position of the result frame.
For example, if the first pixels in the 7 input frames had the values 1.0, 4.4, 2.1, 3.0, 2.0, 4.0, 5.6 then the first pixel in the result frame would be calculated as (1.0 + 2.0 + 2.1)/3 .

`AVERAGE/WEIGHTS av1 = ccdflat.cat ?  -99.9 max,4 0.3,7.23`
As above, but resulting pixel will be the average of the 5 highest pixels.

# AVERAGE/WINDOW

## AVERAGE/WINDOW *core* 06-MAY-1986 KB

**Purpose:** Compute average of (consistent) pixel values

**Syntax:** `AVERAGE/WINDOW out = in_specs [meth] [bgerr,snoise]`

| | |
|---|---|
| `out` | result frame |
| `in_specs` | specification of input frames, either<br>in1,in2,...inj (j < 21) or<br>catalog.cat = name of catalog containing the input frames (at most 20 frames) |
| `meth` | WINDOW, MEDIAN, MAXIMUM or MINIMUM, defaulted to WINDOW |
| `bgerr,snoise` | estimated background error and Gaussian noise only used for meth = WINDOW, defaulted to 0.1,0.1 |

**See also:** AVERAGE/IMAGE, AVERAGE/MEDIAN, AVERAGE/KAPPA, AVERAGE/AVERAGE, AVERAGE/WEIGHTS, AVERAGE/COLUMN, AVERAGE/ROW

**Note:** Except the method WINDOW, all the functionality of AVERAGE/WINDOW is also provided by the more flexible command AVERAGE/IMAGE.
Method = WINDOW:
The output image will contain the flux in units of the exposure time. The background is subtracted from the images before the flux is computed. Only the overlapping area of all input frames will be compared.
The following descriptors must exist for each input frame:
real LHCUTS(5),(6) - low and high limit for valid pixels
double prec. O_TIME(7) - exposure time in seconds (defaulted to 1.)
real FLAT_BKG(1) - background value. (defaulted to 0.)
The pixels of the input frames are compared with the median value (over all input frames). A pixel is rejected, if its uncertainty does not overlap with the median value. The formula for the uncertainty of the flux is
SQRT( bgerr**2 + (v-backgr)*snoise**2 ) / expo_time .
Where v is the pixel value and backgr the background. If all pixels at a certain (x,y) coordinate are rejected, the corresponding pixel of the first input image is used.
Statistics of rejected pixels will be shown after the comparison.

Method = MEDIAN, MAXIMUM or MINIMUM:
The output image will contain the median, maximum or minimum of all the input frames.

**Examples:**
```
AVERAGE/WINDOW good = ccd001,ccd002,ccd008 WINDOW 0.2,0.1
```
Use the frames 'ccd001.bdf', 'ccd002.bdf' and 'ccd08.bdf' as input.

```
AVERAGE/WINDOW darkres = dark.cat MEDIAN
```
Use all frames with an entry in the catalog 'dark.cat' (but max. 20 frames) as input.

## BLINK/CHANNEL *core* 30-JUN-1994 KB

**Purpose:** Blink between Image Display channels.

**Syntax:** `BLINK/CHANNEL [cha1,cha2,..] [intval]`

| | |
|---|---|
| `cha1,cha2,..` | string with at least two channels and up to 4 channels; defaulted to 0,1 |
| `intval` | time in seconds to wait before switching from one channel to the next; defaulted to 0.1 (= 1/10 second) |

**See also:** DISPLAY/CHANNEL, [ImageDisplay]

**Note:** If you are NOT working with an XWindow display, use CTRL/C (CTRL + C) to terminate.
In the X-Window environment place the mouse in the display window and press the Exit button.
If you enter 0.0 as delay, blinking is done continuously.

**Examples:** `BLINK/CHANNEL`
Blink between channel 0 and channel 1 with a delay of 0.1 sec.

`BLINK/CHANNEL 2,1,3 0.2`
Blink between channel 2, 1 and 3 with a delay of 0.2 sec.

## BYE *core* 06-DEC-1993 KB

**Purpose:** Terminate a MIDAS session + return to the host system

**Syntax:** `BYE [proc]`

| | |
|---|---|
| `proc` | name of optional Midas procedure you want to execute before leaving the Midas environment; defaulted to logout |

**See also:** Chapter 3 of MIDAS Users Manual, volume A
host system commands: SETMIDAS, INMIDAS, GOMIDAS (for VMS)
host system commands: $setmidas, $inmidas, $gomidas (for Unix)

**Note:** After the last command in the user procedure 'proc' MIDAS is terminated automatically. I.e. in that procedure you don't need another command BYE.
If you have a procedure named 'logout.prg' in your MID_WORK directory, that procedure is executed whenever you get out of Midas via BYE.

**Examples:** `BYE`
Terminate Midas. If there is a procedure 'logout.prg' in MID_WORK it is executed before exiting Midas.

`BYE lichterfelde`
Execute procedure 'lichterfelde.prg' and terminate Midas.

# CENTER/GAUSS

## CENTER/GAUSS *core* 15-MAR-1994 KB

**Purpose:** Computes the central position of the specified object by fitting a Gaussian to the marginal distributions in both the x- and y-directions (or just in x).

**Subject:** Position, fitting, centering.

**Syntax:** `CENTER/GAUSS [in_specs] [out_specs] [out_opt] [curs_specs]`

`[wsize] [zw_option] [invert_flag]`

`in_specs` input specifications, either
(a) CURSOR - if the subimages are interactively chosen via the cursor rectangle, or
(b) GCURSOR - if the subimages are interactively chosen via the graphic cursor, or
(c) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND, or only columns :XSTART and :XEND (1-dim centering) or only single column :X_POSITION (two rows per interval needed);
(d) image - if subimage will be a (max) 200 * 200 window around center of image defaulted to CURSOR

`out_specs` output specifications:
(e) table name, may be same as given in in_specs
(f) if same table, only the xcenter, ycenter, xerr, yerr, xsigma, ysigma, xfwhm, yfwhm, icent and status will be stored in columns
:XCEN, :YCEN, :XERR, :YERR, :XSIG, :YSIG, :XFWHM :YFWHM :ICENT :STATUS
if new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND, :YEND and :NO will be filled
(g) descriptor,D if the results should be stored in a real descriptor of the involved image frame;
data will be stored as xstart, ystart, xend, yend, xcenter, ycenter, xerr, yerr, xsigma, ysigma, return_status, xfwhm, yfwhm and icent
Per Default the values are displayed only on the terminal and stored in the keyword OUTPUTR which is filled as follows: xstart, ystart, xend, yend, xcenter, ycenter, xerr, yerr, xsigma, ysigma, return_status, xfwhm, yfwhm, icent

`out_opt` A, append data to a descriptor, else start at the beginning, only applicable to (e)
ID, write also column labeled :IDENT to output table, only applicable to (d)
EMISSION, assumes emission lines, applicable to (b)
ABSORPTION, assumes absorption lines, applicable to (b)

`curs_specs` no_curs,drawflag,max_input:
no_curs = 1 or 2, if single cursor a fixed subimage according to next parameter 'wsize' is used; for 2 cursors the size of subimage is determined by the size of the cursor rectangle.
drawflag = option for drawing subwindow limits in the overlay channel;
max_input = max. of cursor inputs.
This parameter is only used if in_specs = CURSOR (a);
defaulted to 2,1,9999

`wsize` no. of x-and y-pixels for subwindow centered via the single cursor;
defaulted to 50,50

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

`invert_flag` YES or NO;
if the objects are at a minimum and not maximum of the data, i.e. instead of a Gaussian 'peak' you have a Gaussian 'valley', set this par. to YES.
defaulted to NO

**See also:** CENTER/MOMENT, CENTER/IQE, MAGNITUDE/CENTER

**Note:** When using the cursor input option, a cursor window will appear on the screen.
For X-Windows use the mouse to move the complete window (or single cursor), use the arrow keys to change window size. Left mouse button (or RETURN key) is ENTER button, right button is EXIT button.
If you also use the 'zw_option', the cursor in the main display is only used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the graphic cursor option (b), the command needs two cursor positions defining the 1D interval to be used. Any key can be used to input the cursor. Use SPACE-BAR to exit.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
The return status from the centering algorithm should be 0 otherwise something is wrong (maybe just some non-convergence)
If you also need the angle of the 2-dim Gaussian (if it's not perfectly round) of the major axis with the X-axis, use the command CENTER/IQE (ImageQualityEstimate).

**Examples:**

`1) CENTER/GAUSS`
Use cursor to define subimages and display start- and end-coordinates of subimage as well as center values and associated errors and return status on terminal only

`2) CENTER/GAUSS CURSOR values,d`
as 1) but write data to beginning of real descriptor VALUES of displayed image

`3) CENTER/GAUSS cursor VALUES,D A`
as 2) but append data to the end of real descriptor VALUES

`4) CENTER/GAUSS CURSOR values`
as 1) but store data into table values.tbl (will be created by MIDAS)

`6) CENTER/GAUSS ccd001,sources`
use columns :XSTART, :YSTART, :XEND and :YEND of sources.tbl to define the subimages of image ccd001.bdf; just display results

`7) CENTER/GAUSS ccd001,sources VALUES,D`
as 6) but write results into real descr values of frame ccd001.bdf

`8) CENTER/GAUSS ccd001,sources sources`
as 6) but store results also in input table sources.tbl

`9) CENTER/GAUSS ccd001,sources center`
as 8) but store results into new table center.tbl

`10) CENTER/GAUSS cursor p6=w`

# CENTER/GAUSS

as 1) but use zoom window to determine working area via cursor, but we need X11 for that...

`11) CENTER/GAUSS GCURSOR table EMISSION`
generate table table.tbl with positions of emission lines defined with the graphic cursor.

CENTER/IQE *core* 21-MAR-1995 KB

**Purpose:** Computes the central position of the specified object by fitting a 2-dim Gaussian to the marginal distributions in both the x- and y-directions (or just in x). Also determine angle of major axis with X-axis.

**Subject:** Fitting, centering, image quality estimation.

**Syntax:** `CENTER/IQE [in_specs] [out_specs] [out_opt] [curs_specs]`

`[wsize] [zw_option] [invert_flag]`

`in_specs` input specifications, either
(a) CURSOR - if the subimages are interactively chosen via the cursor rectangle, or
(c) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND, or only columns :XSTART and :XEND (1-dim centering) or only single column :X_POSITION (two rows per interval needed);
(d) image - if subimage will be a (max) 200 * 200 window around center of image
defaulted to CURSOR

`out_specs` output specifications:
(e) table name, may be same as given in in_specs
(f) if same table, only the xcenter, ycenter, xerr, yerr, xsigma, ysigma, maj.axis, min.axis, icent, return_status and angle, angle_sigma will be stored in columns :XCEN, :YCEN, :XERR, :YERR, :XSIG, :YSIG, :AX_MAJ, :AX_MIN, :ICENT, :STATUS, :ANGLE, :ANGLE_SIG
if new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND, :YEND and :IDENT will be filled
(g) descriptor,D if the results should be stored in a real descriptor of the involved image frame;
data will be stored as xstart, ystart, xend, yend, xcenter, ycenter, xerr, yerr, xsigma, ysigma, return_status, maj.axis, min.axis, angle, angle_sigma and icent
Per Default the values are displayed only on the terminal and stored in the keyword OUTPUTR which is filled as follows: xstart, ystart, xend, yend, xcenter, ycenter, xerr, yerr, xsigma, ysigma, return_status, maj.axis, min.axis, angle, angle_sigma and icent (16 values)
return_status is the status returned from the centering algorithm

`out_opt` A, append data to a descriptor (output table), else start at the beginning (create new table), only applicable to (g), (e)
ID, write also column labeled :IDENT to output table, only applicable to (e)

`curs_specs` no_curs,drawflag,max_input:
no_curs = 1 or 2, if single cursor a fixed subimage according to next parameter 'wsize' is used; for 2 cursors the size of subimage is determined by the size of the cursor rectangle.
drawflag = option for drawing subwindow limits in the overlay channel;
max_input = max. of cursor inputs.
This parameter is only used if in_specs = CURSOR (a);
defaulted to 2,1,9999

`wsize` no. of x-and y-pixels for subwindow centered via the single cursor;
defaulted to 50,50

# CENTER/IQE

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

`invert_flag` YES or NO;
if the objects are at a minimum and not maximum of the data, i.e. instead of a Gaussian 'peak' you have a Gaussian 'valley', set this par. to YES.
defaulted to NO

**See also:** CENTER/MOMENT, CENTER/GAUSS, MAGNITUDE/CENTER

**Note:** When using the cursor input option, a cursor window will appear on the screen.
For X-Windows use the mouse to move the complete window (or single cursor), use the arrow keys to change window size. Left mouse button (or RETURN key) is ENTER button, right button is EXIT button.
If you also use the 'zw_option', the cursor in the main display is only used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
The return status from the centering algorithm should be 0 otherwise something is wrong (maybe just some non-convergence)
The last results are also stored in keyword OUTPUTR(1 - 16), with the angle + sigma-angle (in degrees) in elements 15 + 16.

**Examples:** `1) CENTER/IQE`
Use cursor to define subimages and display start- and end-coordinates of subimage as well as center values and associated errors and return status on terminal only

`2) CENTER/IQE CURSOR values,d`
as 1) but write data to beginning of real descriptor VALUES of displayed image

`3) CENTER/IQE cursor VALUES,D A`
as 2) but append data to the end of real descriptor VALUES

`4) CENTER/IQE CURSOR values`
as 1) but store data into table values.tbl (will be created by MIDAS)

`6) CENTER/IQE ccd001,sources`
use columns :XSTART, :YSTART, :XEND and :YEND of sources.tbl to define the subimages of image ccd001.bdf; just display results

`7) CENTER/IQE ccd001,sources VALUES,D`
as 6) but write results into real descr values of frame ccd001.bdf

`8) CENTER/IQE ccd001,sources sources`
as 6) but store results also in input table sources.tbl

`9) CENTER/IQE ccd001,sources center`
as 8) but store results into new table center.tbl

`10) CENTER/IQE cursor p6=w`
as 1) but use zoom window to determine working area via cursor, but we need X11 for that...

CENTER/MOMENT *core* 15-MAR-1994 KB

**Purpose:** Compute the central position of the specified object using an intensity weighted first moment of the pixel values.

**Subject:** Position, fitting, centering.

**Syntax:** `CENTER/MOMENT [in_specs] [out_specs] [out_opt] [curs_specs]`

`[wsize] [zw_option] [invert_flag]`

`in_specs` input specifications, either
(a) CURSOR - if the subimages are interactively chosen via the cursor rectangle, or
(b) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND, or only columns :XSTART and :XEND (1-dim centering) or only single column :X_POSITION (two rows per interval needed);
(c) image - if subimage will be a (max) 200 * 200 window around center of image
defaulted to CURSOR

`out_specs` output specifications:
(d) table name, may be same as given in in_specs
(e) if same table, only the xcenter, ycenter, xerr, yerr, xsigma, ysigma and status will be stored in columns
:XCEN, :YCEN, :XERR, :YERR, :XSIG, :YSIG, ;STATUS
if new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND, :YEND and :NO will be filled
(f) descriptor,D if the results should be stored in a real descriptor of the involved image frame;
data will be stored as xstart, ystart, xend, yend, xcenter, ycenter, xerr, yerr, xsigma, ysigma and return_status.
Default is display values only on the terminal.

`out_opt` A, append data to a descriptor, else start at the beginning, only applicable to (e)
ID, write also column labeled :IDENT to output table, only applicable to (d)

`curs_specs` no_curs,drawflag,max_input:
no_curs = 1 or 2, if single cursor a fixed subimage according to next parameter 'wsize' is used; for 2 cursors the size of subimage is determined by the size of the cursor rectangle.
drawflag = option for drawing subwindow limits in the overlay channel;
max_input = max. of cursor inputs.
This parameter is only used if in_specs = CURSOR (a);
defaulted to 2,1,9999

`wsize` no. of x-and y-pixels for subwindow centered via the single cursor;
defaulted to 50,50

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

# CENTER/MOMENT

`invert_flag` YES or NO;
if the objects are at a minimum and not maximum of the data, i.e. instead of a 'peak' you have a 'valley', set this par. to YES.
defaulted to NO

**See also:** CENTER/GAUSS, MAGNITUDE/...

**Note:** When using the cursor input option, a cursor window will appear on the screen.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start calculation on subimage. Set both cursors on to move complete cursor window. Change window size by setting one cursor off, move joystick. To exit set both cursors off and press ENTER .
For X-Windows use the mouse to move the complete window (or single cursor), use the arrow keys to change window size. Left mouse button (or RETURN key) is ENTER button, second left button is EXIT button.
If you also use the 'zw_option', the cursor in the main display is only used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
Last results of center calculations are also written into keyword OUTPUTR.
The return status from the centering algorithm should be 0 otherwise something is wrong (maybe just some non-convergence)

**Examples:** `1) CENTER/MOMENT`
Use cursor to define subimages and display start- and end-coordinates of subimage as well as center values and associated errors and return status on terminal only

`2) CENTER/MOMENT CURSOR values,d`
as 1) but write data to beginning of real descriptor VALUES of displayed image

`3) CENTER/MOMENT cursor VALUES,D A`
as 2) but append data to the end of real descriptor VALUES

`4) CENTER/MOMENT CURSOR values`
as 1) but store data into table values.tbl (will be created by MIDAS)

`6) CENTER/MOMENT ccd001,sources`
use columns :XSTART, :YSTART, :XEND and :YEND of sources.tbl to define the subimages of image ccd001.bdf; just display results

`7) CENTER/MOMENT ccd001,sources VALUES,D`
as 6) but write results into real descr values of frame ccd001.bdf

`8) CENTER/MOMENT ccd001,sources sources`
as 6) but store results also in input table sources.tbl

`9) CENTER/MOMENT ccd001,sources center`
as 8) but store results into new table center.tbl

`10) CENTER/MOMENT cursor p6=w`
as 1) but use zoom window to determine working area via cursor, but we need X11 for that...

## CHANGE/DIRECTORY *core* 16-JUN-1993 KB

**Purpose:** Change the default (current) directory for MIDAS

**Syntax:** `CHANGE/DIRECTORY direc`

`direc` string specifying the new default directory in the host system syntax; if no directory string is entered, the current host default directory is used

**See also:** MIDAS Users Guide, Vol. A, Ch. 3

**Note:** Since MIDAS executes its applications in a child process (subprocess for VMS) which leaves no traces after termination, one cannot simply use the host command "$cd ..." ("$SET DEF ..." in VMS) to change the default directory once you are in a MIDAS session. Instead, you must use the MIDAS command CHANGE/DIRECTORY.

**Examples:** `CHANGE/DIR /home/ns2c/kbanse/test`
On a Unix machine set the MIDAS default directory to " /home/ns2c/kbanse/test ". All MIDAS data files and procedures will be taken from that directory and new files will be created there.

`CHANGE/DIR [KBANSE.DATA.93APR]`
On a VMS machine set the MIDAS default directory to " [KBANSE.DATA.93APR] ".

`CHANGE/DIREC`
Reset MIDAS default directory to current host directory.

## CLEAR/ACAT *core* 03-JUN-1991 KB

**Purpose:** Deactivate the ASCII file catalog.

**Syntax:** `CLEAR/ACAT`

**See also:** SET/ACAT

**Note:** Main usage of CLEAR/ACAT and SET/ACAT is for OUTTAPE/FITS command.

**Examples:** `CLEAR/ACAT`
Deactivate an ASCII file catalog which was enabled/activated previously with the command: SET/ACAT catal_name

## CLEAR/ALPHA *core* 12-OCT-1983 KB

**Purpose:** Clear the alpha-numerics memory of the image display.

**Syntax:** `CLEAR/ALPHA`

**See also:** LABEL/DISPLAY

**Note:** All alpha-numerics information displayed in the alpha-numerics memory (for DeAnza) or in the alpha window (for X11) is cleared.

**Examples:** `CLEAR/ALPHA`

# CLEAR/BACKGROUND

## CLEAR/BACKGROUND *core* 04-AUG-1994 KB

**Purpose:** Put Midas session into "foreground" mode

**Syntax:** `CLEAR/BACKGROUND`

**See also:** SET/BACKGROUND, CONNECT/BACK_MIDAS

**Note:** In foreground mode Midas accepts input typed at the keyboard.
This command is typically sent from another Midas session, since the current Midas session being in background mode does not accept any command typed in by the user.

**Examples:** `CLEAR/BACKGR`
Put current Midas from background into foreground (interactive) mode.

## CLEAR/BUFFER *core* 12-OCT-1983 KB

**Purpose:** Clear the command buffer + reset command number to 1.

**Syntax:** `CLEAR/BUFFER`

**See also:** SET/BUFFER, READ/COMMANDS, WRITE/COMMANDS
paragraph about command repetition in chapter 3 of the MIDAS User's Guide, Volume A.

**Note:** The MIDAS command counter has 3 digits. Therefore, after command number 999, the count is reset to 1.

**Examples:** `CLEAR/BUFFER`

## CLEAR/CHANNEL *core* 12-OCT-1983 KB

**Purpose:** Clear + initialize image memory channel.

**Syntax:** `CLEAR/CHANNEL [chanl]`

| | |
|---|---|
| `chanl` | memory channel no.;<br>defaulted to currently displayed channel |

**See also:** SHOW/CHANNEL, CLEAR/DISPLAY

**Note:** The overlay channel may also be accessed via the string "overlay" instead of it's no. (e.g. 2).

**Examples:** `CLEAR/CHANNEL 0`
Initialize image channel 0.

`CLEAR/CHANNEL over`
Initialize the overlay channel.

## CLEAR/CONTEXT *core* 18-DEC-1992 KB

**Purpose:** Remove all command definitions of given context.

**Syntax:** `CLEAR/CONTEXT [context]`

`context` name of a previously enabled context;
if set to '-all', all enabled contexts are cleared, i.e. all commands defined within these contexts are removed;
if set to '-total', also all interactively created commands are removed;
if omitted, the commands of the last enabled (current) context are deleted

**See also:** SET/CONTEXT, SHOW/CONTEXT, SHOW/COMMANDS, CREATE/COMMAND
HELP [Contexts]

**Note:** Contexts are a handy way to add application packages to MIDAS. See also the MIDAS Users Manual, chapter 3, Vol. A.

**Examples:** `CLEAR/CONTEXT`
Remove all commands which have been created via the last enabled context.

`CLEAR/CONTEXT long`
Remove all commands which have been created via the context 'long', which should have been enabled before.

`CLEAR/CONTEXT -all`
Remove all the commands which have been created via any of the currently enabled contexts.

## CLEAR/CURSOR *core* 12-OCT-1983 KB

**Purpose:** Disable cursors

**Syntax:** `CLEAR/CURSOR`

**See also:** SET/CURSOR

**Note:** This command has no effect on X11 displays.
For peripheral displays, e.g. DeAnza, both cursors are cleared,
to enable first cursor #1 and then cursor #2 exclusively, you have to clear the cursors in use first.

**Examples:** `CLEAR/CURSOR`
Make cursors invisible on image display.

# CLEAR/DISPLAY

## CLEAR/DISPLAY *core* 12-OCT-1983 KB

**Purpose:** Reset image display.

**Syntax:** `CLEAR/DISPLAY`

**See also:** INITIALIZE/DISPLAY, CLEAR/CHANNEL

**Note:** For DeAnza, clear all memory channels, set up last channel as graphics/overlay channel + clear split mode.
For XWindow systems clear the display window and the alpha window.

**Examples:** `CLEAR/DISPLAY`

## CLEAR/FCAT *core* 16-JAN-1990 KB

**Purpose:** Disable automatic catalog functions for fit files

**Syntax:** `CLEAR/FCAT`

**See also:** SET/FCAT

**Note:** When the active fit file catalog is cleared, new fit files are no longer entered automatically to this catalog.

**Examples:** `CLEAR/FCAT`
Deactivate an fit file catalog which was enabled/activated previously with the command: SET/FCAT catal_name

## CLEAR/GRAPHIC *core* 12-Oct-1986 RHW

**Purpose:** Erase the screen of the graphic display

**Subject:** Graphics

**Syntax:** `CLEAR/GRAPHIC`

**See also:** CREATE/GRAPHIC, DELETE/GRAPHIC

**Note:** none

**Examples:** `CLEAR/GRAPHICS`
Clear the screen of the graphic device/window.

## CLEAR/ICAT *core* 16-JAN-1990 KB

**Purpose:** Disable automatic catalog functions for image frames

**Syntax:** `CLEAR/ICAT`

**See also:** SET/ICAT

**Note:** When the active image catalog is cleared, new image frames are no longer entered automatically to this catalog.

**Examples:** `CLEAR/ICAT`
Deactivate an image catalog which was enabled/activated previously with the command: SET/ICAT catal_name

## CLEAR/ITT *core* 12-OCT-1983 KB

**Purpose:** Bypass ITT on display of memory

**Syntax:** `CLEAR/ITT [chanl]`

`chanl` Image display memory (or channel), defaulted to currently displayed channel

**See also:** SET/ITT, LOAD/ITT, MODIFY/ITT, EQUALIZE/HISTOGRAM

**Note:** For an explanation of an ITT see the Help of SET/ITT and Chapter 6 of the MIDAS Users Guide.

**Examples:** `CLEAR/ITT`
Disable the ITT for currently displayed image channel.

## CLEAR/LUT *core* 17-OCT-1983 KB

**Purpose:** Bypass (i.e. do not apply) LUT in screen_segment on image display.

**Syntax:** `CLEAR/LUT [screen_segm]`

`screen_segm` screen segment, (0,..,3)
only used in split screen mode (which is only supported on DeAnza devices!); defaulted to 0

**See also:** SET/LUT, LOAD/LUT, DISPLAY/LUT, MODIFY/LUT

**Note:** See help for SET/SPLIT for info on screen segments in split mode.

**Examples:** `CLEAR/LUT`

# CLEAR/OVERLAY

## CLEAR/OVERLAY *core* 07-SEPT-1986 KB

**Purpose:** Disable graphics/overlay plane of display.

**Syntax:** `CLEAR/OVERLAY`

**See also:** SET/OVERLAY, CLEAR/CHANNEL OV

**Note:** Disable the overlay on currently displayed image channel.
This command does not clear the overlay channel! If you enter SET/OVERLAY all graphics will reappear...
To erase the contents of the overlay channel use instead CLEAR/CHANNEL OVERLAY

**Examples:** `CLEAR/OVERLAY`
Disable display of graphics in the overlay channel/plane.

## CLEAR/SCROLL *core* 04-JUL-1990 KB

**Purpose:** Reset scroll values of image memory/channel.

**Syntax:** `CLEAR/SCROLL [chanl]`

`chanl` image channel; defaulted to current channel

**See also:** SCROLL/CHANNEL

**Note:** Scroll values point to upper left corner of image display, so for a 512*512 display the initial scroll values are 0,511 .

**Examples:** `CLEAR/SCROLL`
Reset scroll values of currently displayed image memory (channel) to upper left corner of display.

## CLEAR/SPLIT *core* 12-OCT-1983 KB

**Purpose:** Disable split screen

**Syntax:** `CLEAR/SPLIT`

**See also:** SET/SPLIT

**Note:** Disable split screen mode and show channel 0.
This command is only implemented for DeAnza display systems (IP8000)!

**Examples:** `CLEAR/SPLIT`

## CLEAR/TCAT *core* 16-JAN-1990 KB

**Purpose:** Disable automatic catalog functions for table files

**Syntax:** `CLEAR/TCAT`

**See also:** SET/TCAT

**Note:** When the active table catalog is cleared, new table files are no longer entered automatically to this catalog.

**Examples:** `CLEAR/TCAT`
Deactivate an table catalog which was enabled/activated previously with the command: SET/TCAT catal_name

## CLEAR/ZOOM *core* 12-OCT-1983 KB

**Purpose:** Clear zoom + reset scroll values of image memory channel.

**Syntax:** `CLEAR/ZOOM [chanl]`

`chanl` image memory (channel) no.; defaulted to currently displayed channel

**See also:** ZOOM/CHANNEL

**Note:** This command has no effect on X11 displays.

**Examples:** `CLEAR/ZOOM 0`
Reset zoom factor to 1 and reset scroll values of channel 0 .

## CLOSE/FILE *core* 20-JUN-1993 KB

**Purpose:** Close an ASCII file which was opened before via the OPEN/FILE command.

**Syntax:** `CLOSE/FILE file_id`

`file_id` file id which was returned by a previous OPEN/FILE command
or '*' to close all files which had been opened via OPEN/FILE

**See also:** OPEN/FILE, READ/FILE, WRITE/FILE

**Note:** No error message is displayed, if file closing did not succeed.

**Examples:** `CLOSE/FILE 7`
Close ASCII file with file id 7.

`CLOSE/FILE *`
Close all files wich had been opened via OPEN/FILE.

`CLOSE/FILE fid`
Close ASCII file wich had its file id stored in integer keyword fid in element 1.

# COMPUTE/AIRMASS

## COMPUTE/AIRMASS *core* 02-SEP-1992 DB

**Purpose:** Calculate airmass (from sec z). There are two options:
a) if a frame is used, the parameters are taken from the descriptors of the frame.
b) Parameters are given directly

**Subject:** Spectroscopy, Photometry, Airmass, Flux Calibration

**Syntax:** a) `COMPUTE/AIRMASS frame [long] [lat]`

b) `COMPUTE/AIRMASS alpha delta ST [exptime] [long] [lat] [date] [UT]`

| | |
|---|---|
| `frame` | name of frame with the following descriptors set:<br>O_TIME/R/1/3 date (year,month,date)<br>O_TIME(5) UT (real hours)<br>O_TIME(7) exposure time (seconds<br>O_POS(1) right ascension (real DEGREES!)<br>O_POS(2) declination (real degrees)<br>(use WRITE/DESC to complete info block if necessary) |
| `alpha` | right ascension (hour,min,sec) |
| `delta` | declination (degree,min,sec) |
| `ST` | local mean sidereal time for start of exposure (decimal hour)<br>(if less than 0, the program will supply it, which is also the default; if .GT. 25, airmass will be calculated for object on meridian, i.e. longitude, UT, and date are meaningless) |
| `exptime` | duration of exposure (seconds), defaulted to 0.0 |
| `long` | EAST longitude of observatory (degree,min,sec) default: -70,43,55.35 (= Schmidt telescope, La Silla) (can be omitted if ST is given), must be negative ! NOTE: Eastern longitudes are used conforming to IAU. |
| `lat` | latitude of observatory (degree,min,sec) default: -29,15,25.8 (= Schmidt telescope, La Silla) |
| `date` | civil date (year,month,day), can be omitted if ST is given |
| `UT` | universal time (hour,min,sec), can be omitted if ST is given |

**Output:** a) Output will be written to the following descriptors:
O_TIME(4) Julian date - 2,400,000.5 (days)
O_TIME(6) local mean sidereal time (hours)
O_AIRM airmass
b) Output is written to the following keywords:
OUTPUTR(1) - airmass, OUTPUTD(19) - ST (in real hours)
OUTPUTD(20) - Julian date
ra = right ascension in HH,MM,SS
dec = declination in DD,MM,SS
sid = sidereal time in HH,MM,SS (DEFAULT: = RA)
at half exposure (sid >= 25 -> sid = ra)

**Note:** All input is of type REAL so that, e.g., the date can also be entered as 1987.767,0,0 or just 1987.767 . However, parameters must not be combined. That is, the example given the fraction of the year may contain only the number of elapsed full days, not also UT. For non-zero exposure times, the airmass is calculated as the weighted average according to the following formula (suggested by P. Stetson, Dominion Astrophyical Observatory Preprint, 1988 September):
AM = (AM(ts) + 4 AM(tm) + AM(te))/6,
where AM denotes airmass, and ts, tm, te are the times of start, middle and end of exposure, respectively.
Eastern longitudes are positive and western counted negative conforming to IAU recommendations.

**Examples:** `COMPUTE/AIRMASS ntt1992`
compute airmass for image ntt1992.bdf obtained at La Silla

`COMPUTE/AIRMASS 00,43,11 03,54,41 23.789`
compute airmass for an object at coordinates alpha = 00h43m11s and delta = 03d54'41" observed at sidereal time 23h47m20.4s

`COMPUTE/AIRMASS 13,24,32 -11,05,47 ? ? ? ? 1965,4,1 5,27,20`
compute airmass, JD, and ST for an observation of Spica obtained on La Silla on April 1st, 1965 at 5:27:20 hrs UT.

# COMPUTE/BARYCORR

## COMPUTE/BARYCORR *core* 10-OCT-1995 DB

**Purpose:** Correct universal times and radial velocities to center of sun or barycenter of solar system .

**Syntax:** a) `COMPUTE/BARYCORR date UT alpha delta [longitude] [latitude]`

b) `COMPUTE/BARYCORR table.tbl [longitude] [latitude]`

c) `COMPUTE/BARYCORR image alpha delta [longitude] [latitude]`

| | |
|---|---|
| `date` | year,month,day |
| `UT` | universal time (hour,min,sec) |
| `alpha` | right ascension (hour,min,sec) |
| `delta` | declination (degree,min,sec) |
| `longitude` | EAST longitude of observatory (degree,min,sec),<br>default: -70,43,55.35 (= Schmidt telescope, La Silla)<br>must be negative ! NOTE: Eastern longitudes are used conforming to IAU. |
| `latitude` | of observatory (degree,min,sec), defaulted to -29,15,25.8 (= Schmidt telescope, La Silla) |
| `table` | name of table with the following columns:<br>:AHR, :MIN, :ASEC right ascension of object<br>:DDEG, :DMIN :DSEC declination of object<br>:YEAR :MONTH :DAY date of observation<br>:UTHR :UTMIN :UTSEC UT of observation |
| `image` | name of image, then the date is taken from double prec. descriptor O_TIME(1,2,3) and the UT from O_TIME(5) of that image |
| `longitude` | EAST longitude of observatory (degree,min,sec) default: -70,43,35.55 (= Schmidt telescope, La Silla) NOTE: Eastern longitudes are used conforming to IAU. |
| `latitude` | of observatory (degree,min,sec), defaulted to -29,15,25.8 (= Schmidt telescope, La Silla) |

**Note:** Reference: P. Stumpff, A&A Suppl. Ser. 41, pp. 1-8 (1980)
All times, dates, and coordinates are of type REAL so that, e.g., the date can also be entered as 1987.787,0,0 or just 1987.767 . However, parameters must not be combined. That is, in the example given, the fraction of the year may contain only the number of elapsed full days, not also UT. The same applies to table input, however all input columns must be present even if they contain only zeros. The minus sign for negative values may occur in any field.
a)+ b) Output: will be written to the following keywords:
OUTPUTD/D/1/1 barycentric correction to time (days)
OUTPUTD/D/2/1 heliocentric correction to time (days)
OUTPUTR/R/1/1 barycentric correction to radial velocity (km/s)
OUTPUTR/R/2/1 heliocentric correction to radial velocity (km/s)

b) Output: the following table columns will be created or updated:
:BCT barycentric correction time
:HCT heliocentric correction time
:EBRV earth's barycentric radial velocity
:EHRV earth's heliocentric radial velocity
:EDV earth's diurnal velocity (already included in EBRV and EHRV)

(Note that the table option is very slow.)
For the ultimate accuracy, it will be necessary to precess (command COMPUTE/PRECESS) the coordinates to the date of the observation.
In order to correct measured RV's and JD's, the corrections need to be added as they are, i.e. taking into account their signs.
Eastern longitudes are positive and western counted negative conforming to IAU recommendations.
To enable automatic distinction between tables and images by the command, you should enter the name with the file type (extension), e.g. name.tbl or name.bdf. If no file type is given we default to a table, i.e. we use 'name.tbl' .

**Examples:** `COMPUTE/BARYCORR obslist.tbl 155,28.3 19,49.6`
Compute corrections to UT's and RV's of observations obtained on Mauna Kea, Hawaii

`COMPUTE/BARY 1987,12,24 1,39,10 6,44,35.8 -16,41,56`
Compute corrections to UT's and RV's of observations of Sirius obtained at La Silla on Xmas eve 1987

`COMPUTE/BARYCORR obsfile.bdf 6,44,35.8 -16,41,56`
Compute corrections to UT's and RV's of observations of Sirius obtained at La Silla; date and UT are taken from descriptor O_TIME of 'obsfile.bdf'.

# COMPUTE/COLUMN

## COMPUTE/COLUMN *core* 06-DEC-1994 KB

**Purpose:** Perform arithmetic operations on columns of images and store result in a column of an image.

**Syntax:** `COMPUTE/COLUMN res_frame.column = expression`

`res_frame.column`
image and column to get result of expression, the column is specified by Ci (or ci), with i in [1,NPIX(1)];
if VMS, a new version of the frame will be created by MIDAS

`expression` expression with up to 10 operands, which may be functions and columns or/and constants in the usual algebraic notation, columns of the res_frame are indicated by a leading "C" or "c";
columns of other frames are indicated via 'frame.Ci'

**Note:** The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2), MOD(a1,a2) and ABS(a), are implemented as in FORTRAN. Except that INT(a) will return the nearest integer of "a" converted to real.
The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" defined by the keyword NULL.
This command does not work with 1-dim frames!
If columns from other frames are involved in the expression, they must have the same no. of pixels per column, i.e. NPIX(2). The no. of columns does not have to be the same.

**See also:** COMPUTE/ROW, COMPUTE/IMAGE

**Examples:** `COMPUTE/COLUMN tiburon.C22 = C100+C44`
Use image frame tiburon.bdf and add the 100th column to the 44th column, store the result into the 22nd column of frame tiburon.bdf .

`COMPUTE/COLUMN ballena.c7 = 99.9*sin(delfin.c125)`
Set the 7th column of frame ballena.bdf to the product of 99.9 and the sine of the contents of column no. 125 of frame delfin.bdf . The no. of pixels per column of ballena.bdf and delfin.bdf must be equal (i.e. same NPIX(2)).

`COMPUTE/COLUMN ballena.c2 = 2.*c27`
Compute 2.0 * 27th column of ballena.bdf and store result into second column of frame ballena.bdf .

## COMPUTE/HISTOGRAM *core* 16-AUG-1984 JDP

**Purpose:** Compute histogram of a table column. The result can be produced as a new frame or as a new table with two columns. In the first case, the sampling step is the bin size, and the starting coordinate is the center of the first bin. In the second case, a table with two columns is created where column #1 has the same label as the input column and contains the center of the bin. The column #2 is labelled :FREQUENCY and contains the frequency in the bin.

**Subject:** Statistics, table

**Syntax:** `COMPUTE/HISTOGRAM output = table column [bin [min [max]]]`

| | |
|---|---|
| `output` | name of one dimensional output frame with histogram. The qualifier /TABLE is used to specify the tabular format. If the output has the qualifier /TABLE, name of table to contain histogram. |
| `table` | input table. |
| `column` | input column specification. |
| `bin` | optional bin size. (Default depends on actual range of date |
| `min max` | define, optionally, the range of the histogram. (Defaults are min. and max. values of column specified) |

**Note:** 1) The optional parameters 'min' and 'max' are the lower limit of the first bin and the upper limit of the last bin, resp., while the values quoted as sampling positions for the output image or included in the first column for tabular output are the center of the bins.

2) Image-to-image or image-to-table histogram transformations are performed by the application procedure @a histogram (see HELP/APPLIC histogram).

**See also:** STATISTICS/TABLE, HELP/APPLIC histogram

**Examples:** `COMPUTE/HISTOGRAM hist = mytable :VALUE`
Output histogram into image hist.bdf

`COMPUTE/HISTOGRAM hist/TAB = mytable :VALUE`
Output histogram into two columns of table hist.tbl .

# COMPUTE/IMAGE

## COMPUTE/IMAGE *core* 05-JAN-1993 KB

**Purpose:** Perform arithmetic operations on constants and images, using world coordinates.

**Subject:** Arithmetic.

**Syntax:** `COMPUTE/IMAGE [outspec =] expression`

`outspec` frame to get result of expression. It will be created with the size of the common area of all images involved in the expression. The result frame will be created using real data format, no matter what data format the individual input frames have.
If "outspec =" is omitted, the expression should contain only numerical constants, the result of the computation is shown on the terminal and stored in real keyword OUTPUTR(1).
If descriptor copying is enabled (which is the default,
see also the command SET/MIDAS_SYSTEM DSCCOPY=..),
all non-standard descriptors of the 1. input frame in the expression are copied to the result frame unless you enter 'frame,dscname' as outspec with dscname the name of a frame in the expression;
then, the non-standard descriptors of that frame are copied instead of the ones of the 1. frame in the expression

`expression` arithmetic expression with up to 21 operands, which may be functions and frames or/and constants in the usual algebraic notation. Images must have same stepsize within 0.0001 of a step and same origins within 0.05 of a step. Computations are done on the common area of all images only!

**Note:** Blanks before and after the "=" sign are required. If the name of frame begins with a digit or contains a special char. like, e.g. '+','-',... the name has to be enclosed in double quotes.
If the result frame has the same name as a frame appearing in the expression, no new frame is created, but the result frame overwritten (also in VMS!). This option should be used with care, since computations are only done on the overlapping area of all frames in the expression!
Furthermore, since no new result frame is created, the final data is reconverted to the original data format when closing the result frame and you could get truncation errors!
The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions
SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ABS(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2) and MOD(a1,a2)
are implemented as in FORTRAN. Except, that INT(a) will return the nearest integer of "a" converted to real. The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" stored in the real keyword NULL.

**See also:** COMPUTE/PIXEL, COMPUTE/ROW, COMPUTE/COLUMN, COMPUTE/XYPLANE
COPY/II, REPLACE/IMAGE, @a func2d, SET/MIDAS_SYS

**Examples:** `COMPUTE/IMA r = sqrt(c+5.-log10(b))+abs(aa)`
Compute resulting image 'r.bdf' from the expression involving the frames 'c.bdf', 'b.bdf' and 'aa.bdf'.
If descriptor copying is enabled, all non-standard descriptors of 'c.bdf' (the 1. frame in the expression) are copied to 'r.bdf'.

`COMPUTE/IMA r,aa = sqrt(c+5.-log10(b))+abs(aa)`

As above, but if descriptor copying is enabled, all non-standard descriptors of 'aa.bdf' are copied to 'r.bdf'.

`COMPUTE/IMA EXP(25./3.4)+SIN(1.1)`
Compute the expression and store result in keyword OUTPUTR(1).

`COMPUTE/IMA 10.3*OUTPUTR(1)+4.5`
Use result from computation above.

`COMPUTE/IMA r = sqrt(c))+"qso+22"`
Compute resulting image 'r.bdf' from the expression involving the frames 'c.bdf' and 'qso+22.bdf'.

`COMPUTE/IMA a = a+LOG10(a)`
Also in VMS just modify frame 'a.bdf', do not create a new file.

# COMPUTE/KEYWORD

COMPUTE/KEYWORD *core* 13-FEB-1992 KB

**Purpose:** Compute values of a keyword or compute special function.

**Syntax:** `COMPUTE/KEYWORD key = arithmetic_expression`

`key` keyword with optional parentheses to indicate which element of the keyword shall receive the result of the calculation. This keyword has to exist already!

`arithmetic_expression` with up to 8 operands, which may be functions or/and keywords or/and constants in the usual algebraic notation.
The operators +, -, *, / and ** are supported for numerical keywords, only // (= concatenation) is supported for character keywords

**See also:** READ/KEYWORD, WRITE/KEYWORD, COPY/KEYWORD, COMPUTE/IMAGE
Chapter 3 of the MIDAS Users Guide, Vol. A

**Note:** Supported functions are:
M$INDEX(arg1,arg2) returns index of char. keyword 'arg2' in char. keyword 'arg1' as integer value (identically to INDEX of FORTRAN 77)
M$INDEXB(arg1,arg2) same as above but search is done backwards starting at the end of the string
M$TIME() returns current time as ASCII string (30 characters)
M$SECS() returns current time as no. of seconds elapsed since 1st Jan. 1970 as an integer (this number is also used as timestamp for all updates of keywords)
M$ABS(arg) returns absolute value of integer/real key 'arg' as integer/real
M$NINT(arg) returns nearest integer of real key 'arg'
M$TSTNO(arg) returns 1/0, if character key 'arg' is a number or not
M$LEN(arg) returns length of operand in character key 'arg'
M$EXIST(arg) returns 1/0, if file contained in character key 'arg' exists or not
M$EXISTK(arg) returns 1/0, if keyword contained in char. key 'arg' exists or not
M$EXISTD(arg1,arg2) returns 1/0, if descriptor contained in char. key 'arg2' of file contained in char. key 'arg1' exists or not
M$EXISTC(arg1,arg2) returns -2, if table contained in char. key 'arg1' doesn't exist, the physical number of the column contained in char. key 'arg2' of the table 'arg1', -1 if the column doesn't exist M$SYMBOL(arg) returns the translation of DCL symbol (VMS) or shell variable (Unix) contained in char. keyword 'arg' as character string
M$LN, M$EXP, M$LOG, M$SIN, M$COS and M$TAN(arg), M$SQRT, M$ASIN, M$ACOS and M$ATAN(arg) return real + double prec. results for 'arg' an integer or real + double prec. keyword (angles are always expressed in degrees)
M$UPPER(arg) returns contents of char. keyword 'arg' in upper case
M$LOWER(arg) returns contents of char. keyword 'arg' in lower case
M$AGL(arg) returns contents of AGL definitions file 'agldevices.dat' related to char. keyword 'arg' as an ASCII string. This function is useful if you have procedures working with different output devices.
M$FILTYP(arg1,arg2) returns a type_no for file contained in char. key 'arg1'. Type_no = 1 for an image, = 2 for a table, = 3 for a fitfile and = 0 otherwise. If the file name does not include a file type, the type definition stored in char. key 'arg2' is appended to the file name. If you want to use a file without file type, set arg2 to " ".
*** As a short form of this command you may omit the COMPUTE/KEY ***

**Examples:** `COMPUTE/KEYWORD INPUTI(3) = 7*OUTPUTI(12)`
Multiply the contents of element 12 of key OUTPUTI by 7 and store the result into element 3 of key INPUTI

`inputi(3) = 7 * outputi(12)`
short form of the command above ...

`COMPUTE/KEYWORD INPUTI = M$INDEX(IN_A,".tbl")`
Test, if key IN_A holds table name

`COMPUTE/KEYWORD INPUTC = M$TIME()`
Store current time in ASCII into char. keyword INPUTC. Note, that the parentheses are necessary.

`COMPUTE/KEYWORD INPUTI(6) = M$EXIST(P1)`
Depending upon if the file, the name of which is stored in key P1, exists or not, INPUTI(6) will be set to 1 or 0.

`INPUTD(3) = M$EXP(INPUTD(10))`
If INPUTD(10) = val, calculate e**val and store it into INPUTD(3)

---

## COMPUTE/PIXEL *core* 05-JAN-1993 KB

**Purpose:** Perform arithmetic operations on images, using pixel coordinates.

**Syntax:** `COMPUTE/PIXEL [outspec =] expression`

| | |
|---|---|
| `outspec` | frame to get result of expression, it will be created by MIDAS with the size of common area of all images involved in the expression<br>If descriptor copying is enabled (which is the default, see also the command SET/MIDAS_SYSTEM DSCCOPY=..), all non-standard descriptors of the 1. input frame in the expression are copied to the result frame unless you enter 'frame,dscname' as outspec with dscname the name of a frame in the expression;<br>then, the non-standard descriptors of that frame are copied instead of the ones of the 1. frame in the expression |
| `expression` | arithmetic expression with up to 21 operands, which may be functions and frames or/and constants in the usual algebraic notation |

**See also:** COMPUTE/IMAGE, REPLACE/IMAGE

**Note:** This command works exactly like COMPUTE/IMAGE but uses pixel coords; i.e. all images involved are treated like arrays ranging from 1 to NPIX in each dimension regardless what stepsize they have.
Computations are done only on the common pixel area of all images.
Consult the help of COMPUTE/IMAGE for a detailed description of all available functions.

**Examples:** `COMPUTE/PIXEL nero = caligula+claudius`
Add contents of image frame 'caligula.bdf' and 'claudius.bdf' and store result into frame 'nero.bdf'.
If descriptor copying is enabled, all non-standard descriptors of 'caligula.bdf' (the 1. frame in the expression) are copied to the result frame 'nero.bdf'.

# COMPUTE/PRECESSION

## COMPUTE/PRECESSION *core* 12-NOV-1993 KB

**Purpose:** Precess equatorial coordinates from one epoch to another.

**Syntax:** `COMPUTE/PRECESSION alpha delta equinox0 equinox1`

`COMPUTE/PRECESSION table.tbl equinox0 equinox1`

| | |
|---|---|
| `alpha` | original right ascension (hour,min,sec) |
| `delta` | original declination (degree,min,sec) |
| `equinox0` | original equinox (year,month,day) |
| `equinox1` | equinox of precessed coordinates (year,month,day) |
| `table` | name of table with the following columns:<br>:AHR, :AMIN, :ASEC right ascension (hour,min,sec)<br>:DDEG, :DMIN, :DSEC declination (degree, min,sec)<br>if 'equinox0' is set to TAB, different original equinoxes apply to the coordinates and must be given in columns :EYEAR, :EMONTH, and :EDATE |

**See also:** COMPUTE/ST, COMPUTE/UT, COMPUTE/TABLE

**Note:** The parameters may also be referenced via:
ALPHA0= DELTA0= EQUINOX0= EQUINOX1=
Dates and coordinates are of type REAL so that, e.g., the date can also be entered as 1982.787,0,0 or just 1982.787 . However, parameters may not be combined. That is, in the example given, the fraction of the year may contain only the number of elapsed full days not also UT.
The same applies to table input, however all input columns must be present even if they contain only zeros. The minus sign for negative values may occur in any field.
Output will be written to the following keywords:
OUTPUTR/R/1/3 precessed right ascension (hour,min,sec)
OUTPUTR/R/4/3 precessed declination (degree,min,sec)
OUTPUTD/D/1/2 original and new epoch
With the table option the following table columns will be created or updated:
:PAHR, :AMIN :PASEC precessed right ascension (hour,min,sec)
:PDDEG, :PDMIN, :PDSEC precessed declination (degree,min,sec)
Also the descriptor NEW_EPOCH will contain the new equinox.
If parameter P2 was set to TAB, a descriptor OLD_EPOCH will contain the original equinox.
(Note that the table option is rather slow.)

**Examples:** `COMPUTE/PRECESS 13,24,32.0 -11,5,47 1987,7,1.333 2000.0`
Precess the coordinates of Spica from 1987 July 1 8:00:00 UT to 2000.0

`COMPUTE/PRECESS stars.tbl 1987,7,1.333 1993.671`
Precess coordinates contained in table 'stars.tbl' from the equinox 1987 July 1 8:00:00 UT to the equinox 1993.671.

`COMPUTE/PRECESS stars.tbl tab 1993.671`
Precess coordinates contained in table 'stars.tbl' to the common equinox 1993.671. All input coordinates may relate to different equinoxes which in any case are given in columns :EYEAR, :EMONTH, and :EDATE of table 'stars.tbl'.

## COMPUTE/REGRESSION *core* 12-OCT-1983 JDP

**Purpose:** Compute the result of a regression. A new column is created or values in a previously existing column are replaced by the fitted values in the regression. Regression coefficients are referred to by the name assigned in a previous SAVE/REG command.

**Subject:** Regression, fitting data

**Syntax:** `COMPUTE/REGRESSION table column = name[(ind)] [d_type]`

| | |
|---|---|
| `table` | table name |
| `column` | output column, either already existing or a new one created by the command. |
| `name` | user's name of the regression coefficients saved by a previous SAVE/REGRESSION command |
| `ind` | optional column containing the independent variable(s). By default the command uses the same column(s) as in the command REGRESSION. |
| `d_type` | defines the data type for the output column as R*4 or R*8. If the column already exists this parameter is not relevant. By default the output type is R*4. |

**Note:** The computed result of the regression can be used to select on small residuals and repeat the REGRESSION command to get better coefficients. The select mask is reset to the complete table before the computation.

**See also:** REGRESSION/TABLE, SAVE/REGRESSION

**Examples:** `The following sequence of commands will compute the result` of a polynomial regression between columns :X and :Y

```
REGRESSION/POLYNOMIAL mytable :Y :X 2

SAVE/REGRESSION mytable test

COMPUTE/REGRESSION mytable :YFIT = test(:X)
```

# COMPUTE/ROW

| COMPUTE/ROW | *core* | 06-DEC-1994 KB |
|---|---|---|

**Purpose:** Perform arithmetic operations on rows of images and store result in a row of an image.

**Syntax:** `COMPUTE/ROW res_frame.row = expression`

`res_frame.row` image and row to get result of expression, the row is specified by Ri (or ri), with i in [1,NPIX(2)];
if VMS, a new version of the frame will be created by MIDAS

`expression` expression with up to 10 operands, which may be functions and rows or/and constants in the usual algebraic notation, rows of the res_frame are indicated by a leading "R" or "r";
rows of other frames are indicated via 'frame.Ri'

**Note:** The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2), MOD(a1,a2) and ABS(a), are implemented as in FORTRAN. Except that INT(a) will return the nearest integer of "a" converted to real.
The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" defined by the keyword NULL.
If rows from other frames are involved in the expression, they must have the same no. of pixels per row, i.e. NPIX(1). The no. of rows does not have to be the same, i.e. you can mix 1-dim and 2-dim frames.

**See also:** COMPUTE/COLUMN, COMPUTE/IMAGE

**Examples:**
```
COMPUTE/ROW tiburon.R22 = R100+R44
```
Use image frame tiburon.bdf and add the 100th row to the 44th row, store the result into the 22nd row of frame tiburon.bdf .

```
COMPUTE/ROW ballena.r7 = 99.9*sin(delfin.r125)
```
Set the 7th row of frame ballena.bdf to the product of 99.9 and the sine of the contents of row no. 125 of frame delfin.bdf . The no. of pixels per row of ballena.bdf and delfin.bdf must be equal.

```
COMPUTE/ROW ballena.r2 = 2.*r27
```
Compute 2.0 * 27th row of ballena.bdf and store result into second row of frame ballena.bdf .

## COMPUTE/ST *core* 02-SEP-1992 DB

**Purpose:** Calculate geocentric Julian date (JD) and local mean sidereal time (ST) from civil date and universal time (UT).

**Syntax:**

```
COMPUTE/ST date UT [longitude]
COMPUTE/ST table.tbl [longitude]
COMPUTE/ST image [longitude]
```

| | |
|---|---|
| `date` | year,month,day of civil date at longitude zero (Greenwich) NOT at the longitude of the observatory! |
| `UT` | hour,min,sec of universal time |
| `longitude` | degree,min,sec of eastern longitude of observatory, defaulted to -70,43,55.35 (= Schmidt telescope, La Silla) must be negative ! NOTE: Eastern longitudes are used conforming to IAU. |
| `table` | name of input table with the following column labels: :YEAR :MONTH :DAY date :UTHR :UTMIN :UTSEC universal time (UT) |
| `image` | name of image with the following descriptors set: O_TIME/D/1/3 year,month,day O_TIME(5) universal time in real hours |

**See also:** COMPUTE/UT, COMPUTE/PRECESSION

**Note:** The parameters may also be referenced via:
DATE= UT= LONGITUDE=
Dates and coordinates are of type REAL so that, e.g., the date can also be entered as 1982.787,0,0 or just 1982.787 . However, parameters must not be combined. That is, in the example given, the fraction of the year may contain only the number of elapsed full days not also UT.
The same applies to table input, however all input columns must be present even if they contain only zeros. The minus sign for negative values may occur in any field.
a) output keywords:
OUTPUTR/R/1/3 local mean sidereal time (hour,min,sec)
OUTPUTD/D/1/1 Julian date
b) for the table option the following table columns will be created or updated:
:STHR :STMIN :STSEC - local mean ST
:JD - Julian date
(Note that the table option is rather slow.)
c) the following descriptors will be created or updated:
O_TIME(4) Julian date -2,400,000.5 in days
Eastern longitudes are positive and western counted negative conforming to IAU recommendations.

**Examples:**

```
COMPUTE/ST ccd0007
```
Compute JD and ST for image 'ccd0007.bdf' obtained at La Silla

```
COMPUTE/ST obslist.tbl 0,0,0
```
Compute JD and ST for objects in table 'obslist.tbl', done in preparation for observations at the Royal Observatory in Greenwich.

```
COMPUTE/ST 1987,5,7 3,17,30
```
Compute JD for 03:17:30 hrs UT on 7 May 1987 and the local mean sidereal time at La Silla

# COMPUTE/TABLE

## COMPUTE/TABLE *core* 05-Aug-1992 MP

**Purpose:** Perform arithmetic or string operations on columns in a table. The output column will be created if it doesn't exist.

**Subject:** Table, column, arithmetic, string

**Syntax:** `COMPUTE/TABLE table column = expression`

| | |
|---|---|
| `table` | table name |
| `column` | output column, either already existing or a new one created by the command (In that case, the column has to be referred to by label) . |
| `expression` | FORTRAN compatible expression , in which the variables are columns in the table. |

**Note:** Expression may contain arithmetic operators, logical operators and mathematical functions. The supported arithmetic operators are +, -, *, / and **. The supported mathematical functions are :
SQRT(:a) LN(:a) LOG10(:a) EXP(:a) SIN(:a)
COS(:a) TAN(:a) ASIN(:a) ACOS(:a) ATAN(:a)
SINH(:a) COSH(:a) TANH(:a) ABS(:a) INT(:a)
MIN(:a,:b) MAX(:a,:b) MOD(:a,:b)
Arguments of trigonometric functions are in degrees. The function INT produces the nearest integer value. The function MOD returns the floating-point remainder of a/b with the same sign as a.
The supported logical operators are .AND. .OR. .NOT.

String operations can be performed on character columns:
TOLOWER(:a) convert all characters in the column to uppercase TOUPPER(:a) convert all characters in the column to lowercase COLLAPSE(:a) remove all spaces from the contents of the column CONCAT(:a,:b) CONCAT(:a,s) concatenate the contents of two columns or of one column and one string into an output column

The function TOCHAR(:a) can be used to convert the values of a column with any numerical type into strings, using the output FORTRAN format associated to the input column.

The variable name SEQUENCE (short form SEQ) can be used as variable, refering to the sequence number of the table entries.
The variable name SELECT (short form SEL) can be used as variable, refering to the select flag of the table entries (The value will be 0 or 1 according to whether or not the entry is selected). This variable can be used to perform computations only on the selected entries of the table.

If the output column doesn't exist, it will be created and its type will be the same as the "highest" one of the input columns.
The result of an illegal operation (i.e logarithm of a negative number) will be set to the NULL value.
The SELECT mask will be reset by this command, i.e., before the computation, all the elements in the table are selected.

**See also:** CREATE/COL, WRITE/TABLE, SELECT/TAB

**Examples:**
```
COMPUTE/TAB mytable :MAG = 21.3-2.5*LOG10(:INTEN-:BACK)
```
– compute resulting column :MAG from the expression involving the input columns :INTEN and :BACK
```
COMPUTE/TAB mytable :R = 0.5*SEQ
```

– compute resulting column :R from the expression involving the sequence number of the table entries.

```
SELECT/TAB mytable :MAG.LT.110.
```

COMPUTE/TAB mytable :Z = MIN(:MAG,:R)+SEL – compute resulting column :Z as follows: compute for each row the minimum value of the column :MAG and the column :R and add to the result the selection flag of the row

```
COMPUTE/TAB mytable :NAME = CONCAT(:STAR,TOCHAR(:SEQ))
```

– For each row of the table myname, convert the value of the column :SEQ into a string, concatenate it to the contents of the column :STAR and store the result into the column :NAME

# COMPUTE/UT

## COMPUTE/UT *core* 02-SEP-1992 DB

**Purpose:** Calculate geocentric Julian date (JD) and universal time (UT) from civil date and local mean sidereal time (ST)

**Syntax:** `COMPUTE/UT date ST [longitude]`

`COMPUTE/UT table.tbl [longitude]`

`COMPUTE/UT image [longitude]`

| | |
|---|---|
| `date` | year,month,day of civil date |
| `ST` | hour,min,sec of sidereal time |
| `longitude` | degree,min,sec of eastern longitude of observatory,<br>defaulted to: -70,43,55.35<br>(= Schmidt telescope, La Silla), must be negative !<br>NOTE: Eastern longitudes are used conforming to IAU. |
| `table` | name of input table with the following column labels:<br>:YEAR :MONTH :DAY date<br>:STHR :STMIN :STSEC sidereal time (ST) |
| `image` | name of image which must have the following descriptors<br>O_TIME/D/1/3 year,month,day<br>O_TIME(6) local mean ST in real hours |

**See also:** COMPUTE/ST, COMPUTE/PRECESSION

**Note:** The parameters may also be referenced via:
DATE= ST= LONGITUDE= .
Dates and coordinates are of type REAL so that, e.g., the date can also be entered as 1982.787,0,0 or just 1982.787 . However, parameters must not be combined. That is, in the example given, the fraction of the year may contain only the number of elapsed full days not also UT.
The same applies to table input, however all input columns must be present even if they contain only zeros. The minus sign for negative values may occur in any field.

a) output keyword:
OUTPUTR/R/1/3 universal time (hour,min,sec)
OUTPUTD/D/1/1 Julian date
b) for the table option the following table columns will be created or updated:
:UTHR :UTMIN :UTSEC - UT
:JD - Julian date
(Note that the table option is rather slow.)
c) the following descriptors will be created or updated:
O_TIME(4) - Julian date -2,400,000.5 in days
O_TIME(5) universal time in real hours
Eastern longitudes are positive and western counted negative conforming to IAU recommendations.

**Examples:** `COMPUTE/UT ccd0007`
Compute JD and UT for image 'ccd0007.bdf' obtained at La Silla.

`COMPUTE/UT obslist.tbl 0,0,0`
Compute JD and UT for objects in table 'obslist.tbl' done in preparation for observations at the Royal Observatory in Greenwich.

`COMPUTE/UT 1987,5,7 15,17,30`
Compute JD and UT for 15:17:30 hrs La Silla mean ST on 7 May 1987.

COMPUTE/WEIGHTS *core* 24-JAN-1994 KB

**Purpose:** Determine weights for command AVERAGE/WEIGHTS. A window (usually of the background) will be averaged over all input images. The weights will be computed acoording to deviation in each window from the mean.

**Syntax:** `COMPUTE/WEIGHTS input_specs [window_specs]`

`input_specs` ima1,ima2,... or imacatal.cat to specify the input images (imacatal.cat the name of the image catalog containing all input images)

`window_specs` CURSOR or xsta,ysta,xend,yend to indicate the corners of the window we work in;
xsta,ysta,xend,yend as pixel numbers (@...) or world coords. of the window;
defaulted to CURSOR

**Note:** For the CURSOR option, one of the input images must be displayed, so that the window can be determined interactively with the cursor rectangle.
The resulting weights will be displayed on the terminal and stored in a real descriptor WEIGHT (of 1 element) for each input frame.
This command does not work for 1-dim frames, however you can specify 1-dim windows!

**See also:** AVERAGE/WEIGHTS

**Examples:** `COMPUTE/WEIGHTS n1068,n1069,n1070 10.3,10.3,2000.,2000.`
Use a square window on all 3 input frames.

`COMPUTE/WEIGHTS galax023.cat`
Work on all images contained in the image catalog 'galax023.cat'. One of these images must be currently displayed, so that we can specify the window with the cursor rectangle interactively.

# COMPUTE/XYPLANE

## COMPUTE/XYPLANE *core* 12-OCT-1995 KB

**Purpose:** Compute arithmetic expressions on the xy_planes of image cubes.

**Subject:** Arithmetic.

**Syntax:** `COMPUTE/XYPLANE result_cube = expression`

`result_cube` frame to get result of expression. The result frame will be created using real data format, no matter what data format the individual input frames have.
If descriptor copying is enabled (which is the default, see also the command SET/MIDAS_SYSTEM DSCCOPY=..),
all non-standard descriptors of the first 3-dim input frame in the expression are copied to the result frame unless you enter 'frame,dscname' as result_cube with dscname the name of a frame in the expression;
then, the non-standard descriptors of that frame are copied instead of the ones of the first 3-dim frame in the expression

`expression` arithmetic expression with up to 21 operands, which may be functions and frames or/and constants in the usual algebraic notation

**Note:** The xy_plane of a cube is formed by the x- and y-axis and has NPIX(1) pixels per row and NPIX(2) pixels per column.
If a frame operand has NAXIS=2, the single plane of that frame is used in the computation for all xy_planes of the result frame.
If a frame operand has NAXIS=1, the single line of that frame is used in the computation for all lines of all xy_planes of the result frame.
There must be at least one 3-dim frame operand in the arithmetic expression, from which the dimensions of the result frame are taken over.
If the result frame has the same name as a frame appearing in the expression, no new frame is created, but the result frame is overwritten. Furthermore, since no new result frame is created, the final data is reconverted to the original data format when closing the result frame and you could get truncation errors!
The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions
SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ABS(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2) and MOD(a1,a2)
are implemented as in FORTRAN. Except, that INT(a) will return the nearest integer of "a" converted to real. The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" stored in the real keyword NULL.

**See also:** COMPUTE/XZPLANE, COMPUTE/ZYPLANE, COMPUTE/IMAGE
TRANSPOSE/CUBE, SET/MIDAS_SYS

**Examples:**
```
COMPUTE/XYPLANE res3 = ima3+10*jma3
```
Compute the xy-planes of the resulting cube 'res3.bdf' from the expression combining the xy-planes of the cubes 'ima3.bdf' and 'jma3.bdf'. The size and no. of xy-planes of the cubes 'ima3.bdf' and 'jma3.bdf' must be equal.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.
If descriptor copying is enabled, also all non-standard descriptors of 'ima3.bdf' (the 1. 3-dim frame in the expression) are copied to 'res3.bdf'.

```
COMPUTE/XYPLANE res3,jma3 = ima3+10*jma3
```

As above, but if descriptor copying is enabled, all non-standard descriptors of 'jma3.bdf' are copied to 'res3.bdf'.

`COMPUTE/XYPLANE res3 = ima2-ima3`

Compute the xy-planes of the resulting cube 'res3.bdf' from the difference of the 2-dim frame 'ima2.bdf' and all xy-planes of the cube 'ima3.bdf'. The size of 'ima2.bdf' and the size of the xy-planes of 'ima3.bdf' must be equal. That means, NPIX(1) and NPIX(2) must be equal for 'ima2.bdf' and 'ima3.bdf'.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

`COMPUTE/XYPLANE res3 = ima2-ima3+sin(ima1)`

Compute the xy-planes of the resulting cube 'res3.bdf' from the expression involving the 2-dim frame 'ima2.bdf', the 3-dim frame 'ima3.bdf' and the 1-dim frame 'ima1.bdf'. The 2-dim frame 'ima2.bdf' is used with all xy-planes of the cube 'ima3.bdf'. Also, the 1-dim frame 'ima1.bdf' is used with all lines of 'ima2.bdf' and all lines of all xy-planes of 'ima3.bdf'. The size of 'ima2.bdf' and the size of the xy-planes of 'ima3.bdf' must be equal. Also, the no. of pixels of 'ima1.bdf' must be equal to the no. of pixels per line of the xy-planes of 'ima3.bdf'. That means, NPIX(1) must be equal for all frames, and NPIX(2) must be equal for 'ima2.bdf' and 'ima3.bdf'.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

# COMPUTE/XZPLANE

## COMPUTE/XZPLANE *core* 12-OCT-1995 KB

**Purpose:** Compute arithmetic expressions on the xz_planes of image cubes.

**Subject:** Arithmetic.

**Syntax:** `COMPUTE/XZPLANE result_cube = expression`

`result_cube` frame to get result of expression. The result frame will be created using real data format, no matter what data format the individual input frames have.
If descriptor copying is enabled (which is the default, see also the command SET/MIDAS_SYSTEM DSCCOPY=..),
all non-standard descriptors of the first 3-dim input frame in the expression are copied to the result frame unless you enter 'frame,dscname' as result_cube with dscname the name of a frame in the expression;
then, the non-standard descriptors of that frame are copied instead of the ones of the first 3-dim frame in the expression

`expression` arithmetic expression with up to 21 operands, which may be functions and frames or/and constants in the usual algebraic notation

**Note:** The xz_plane of a cube is formed by the x- and z-axis and has NPIX(1) pixels per row and NPIX(3) pixels per column.
If a frame operand has NAXIS=2, the single plane of that frame is used in the computation for all xz_planes of the result frame.
If a frame operand has NAXIS=1, the single line of that frame is used in the computation for all lines of all xz_planes of the result frame.
There must be at least one 3-dim frame operand in the arithmetic expression, from which the dimensions of the result frame are taken over.
If the result frame has the same name as a frame appearing in the expression, no new frame is created, but the result frame is overwritten. Furthermore, since no new result frame is created, the final data is reconverted to the original data format when closing the result frame and you could get truncation errors!
The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions
SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ABS(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2) and MOD(a1,a2)
are implemented as in FORTRAN. Except, that INT(a) will return the nearest integer of "a" converted to real. The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" stored in the real keyword NULL.

**See also:** COMPUTE/XYPLANE, COMPUTE/ZYPLANE, COMPUTE/IMAGE
TRANSPOSE/CUBE, SET/MIDAS_SYS

**Examples:**
```
COMPUTE/XZPLANE res3 = ima3+10*jma3
```
Compute the xz-planes of the resulting cube 'res3.bdf' from the expression combining the xz-planes of the cubes 'ima3.bdf' and 'jma3.bdf'. The size and no. of xz-planes of the cubes 'ima3.bdf' and 'jma3.bdf' must be equal.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.
If descriptor copying is enabled, also all non-standard descriptors of 'ima3.bdf' (the 1. 3-dim frame in the expression) are copied to 'res3.bdf'.

```
COMPUTE/XZPLANE res3,jma3 = ima3+10*jma3
```

As above, but if descriptor copying is enabled, all non-standard descriptors of 'jma3.bdf' are copied to 'res3.bdf'.

`COMPUTE/XZPLANE res3 = ima2-ima3`

Compute the xz-planes of the resulting cube 'res3.bdf' from the difference of the 2-dim frame 'ima2.bdf' and all xz-planes of the cube 'ima3.bdf'. The size of 'ima2.bdf' and the size of the xz-planes of 'ima3.bdf' must be equal. That means, NPIX(1),(2) of 'ima2.bdf' and NPIX(1),(3) of 'ima3.bdf' must be equal.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

`COMPUTE/XZPLANE res3 = ima2-ima3+sin(ima1)`

Compute the xz-planes of the resulting cube 'res3.bdf' from the expression involving the 2-dim frame 'ima2.bdf', the 3-dim frame 'ima3.bdf' and the 1-dim frame 'ima1.bdf'. The 2-dim frame 'ima2.bdf' is used with all xz-planes of the cube 'ima3.bdf'. Also, the 1-dim frame 'ima1.bdf' is used with all lines of 'ima2.bdf' and all lines of all xz-planes of 'ima3.bdf'. The size of 'ima2.bdf' and the size of the xz-planes of 'ima3.bdf' must be equal. Also, the no. of pixels of 'ima1.bdf' must be equal to the no. of pixels per line of the xz-planes of 'ima3.bdf'. That means, NPIX(1) must be equal for all frames, and NPIX(2) of 'ima2.bdf' must be equal to NPIX(3) of 'ima3.bdf'.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

# COMPUTE/ZYPLANE

## COMPUTE/ZYPLANE *core* 12-OCT-1995 KB

**Purpose:** Compute arithmetic expressions on the zy_planes of image cubes.

**Subject:** Arithmetic.

**Syntax:** `COMPUTE/ZYPLANE result_cube = expression`

`result_cube` frame to get result of expression. The result frame will be created using real data format, no matter what data format the individual input frames have.
If descriptor copying is enabled (which is the default, see also the command SET/MIDAS_SYSTEM DSCCOPY=..),
all non-standard descriptors of the first 3-dim input frame in the expression are copied to the result frame unless you enter 'frame,dscname' as result_cube with dscname the name of a frame in the expression;
then, the non-standard descriptors of that frame are copied instead of the ones of the first 3-dim frame in the expression

`expression` arithmetic expression with up to 21 operands, which may be functions and frames or/and constants in the usual algebraic notation

**Note:** The zy_plane of a cube is formed by the z- and y-axis and has NPIX(3) pixels per row and NPIX(2) pixels per column.
If a frame operand has NAXIS=2, the single plane of that frame is used in the computation for all zy_planes of the result frame.
If a frame operand has NAXIS=1, the single line of that frame is used in the computation for all lines of all zy_planes of the result frame.
There must be at least one 3-dim frame operand in the arithmetic expression, from which the dimensions of the result frame are taken over.
If the result frame has the same name as a frame appearing in the expression, no new frame is created, but the result frame is overwritten. Furthermore, since no new result frame is created, the final data is reconverted to the original data format when closing the result frame and you could get truncation errors!
The operations +, -, *, / and ** are supported with the same precedence as in FORTRAN. Parentheses may be used to change that order as well as to nest operations.
The functions
SQRT(a), EXP(a), EXP10(a), LN(a), LOG10(a), SIN(a) ASIN(a), COS(a), ACOS(a), TAN(a), ATAN(a), INT(a), ABS(a), ATAN2(a1,a2), MAX(a1,a2), MIN(a1,a2) and MOD(a1,a2)
are implemented as in FORTRAN. Except, that INT(a) will return the nearest integer of "a" converted to real. The trigonometric functions expect arguments in degrees!
Results of illegal operations (e.g. division by zero) are set to the "null value" stored in the real keyword NULL.

**See also:** COMPUTE/XYPLANE, COMPUTE/XZPLANE, COMPUTE/IMAGE
TRANSPOSE/CUBE, SET/MIDAS_SYS

**Examples:** `COMPUTE/ZYPLANE res3 = ima3+10*jma3`
Compute the zy-planes of the resulting cube 'res3.bdf' from the expression combining the zy-planes of the cubes 'ima3.bdf' and 'jma3.bdf'. The size and no. of zy-planes of the cubes 'ima3.bdf' and 'jma3.bdf' must be equal.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.
If descriptor copying is enabled, also all non-standard descriptors of 'ima3.bdf' (the 1. 3-dim frame in the expression) are copied to 'res3.bdf'.

```
COMPUTE/ZYPLANE res3,jma3 = ima3+10*jma3
```

As above, but if descriptor copying is enabled, all non-standard descriptors of 'jma3.bdf' are copied to 'res3.bdf'.

`COMPUTE/ZYPLANE res3 = ima2-ima3`

Compute the zy-planes of the resulting cube 'res3.bdf' from the difference of the 2-dim frame 'ima2.bdf' and all zy-planes of the cube 'ima3.bdf'. The size of 'ima2.bdf' and the size of the zy-planes of 'ima3.bdf' must be equal. That means, NPIX(1),(2) of 'ima2.bdf' and NPIX(3),(2) of 'ima3.bdf' must be equal.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

`COMPUTE/ZYPLANE res3 = ima2-ima3+sin(ima1)`

Compute the zy-planes of the resulting cube 'res3.bdf' from the expression involving the 2-dim frame 'ima2.bdf', the 3-dim frame 'ima3.bdf' and the 1-dim frame 'ima1.bdf'. The 2-dim frame 'ima2.bdf' is used with all zy-planes of the cube 'ima3.bdf'. Also, the 1-dim frame 'ima1.bdf' is used with all lines of 'ima2.bdf' and all lines of all zy-planes of 'ima3.bdf'. The size of 'ima2.bdf' and the size of the zy-planes of 'ima3.bdf' must be equal. Also, the no. of pixels of 'ima1.bdf' must be equal to the no. of pixels per line of the zy-planes of 'ima3.bdf'. That means, NPIX(1) of 'ima2.bdf' as well as of 'ima1.bdf' must be equal to NPIX(3) of 'ima3.bdf'. And NPIX(2) of 'ima2.bdf' must be equal to NPIX(2) of 'ima3.bdf'.
The dimensions of 'res3.bdf' are taken from 'ima3.bdf'.

# CONNECT/BACK_MIDAS

## CONNECT/BACK_MIDAS *core* 27-OCT-1994 KB

**Purpose:** Connect "command syntax" to another MIDAS.

**Syntax:** `CONNECT/BACK_MIDAS unit wait_specs b_char method`

| | |
|---|---|
| `unit` | unit of Midas session we want to connect to; |
| `wait_specs` | 'nowait' or 'wait[,max_wait]' to indicate, if we want to wait until the background command terminates or not;<br>if ',max_wait' is given, at most max_wait seconds are waited, then control is returned to the current MIDAS |
| `b_char` | character to indicate a background command;<br>'bchar,command' will execute the command in the Midas session with 'unit' given above |
| `method` | FILES or SOCKETS to define the mechanism used for the communication with the "background" MIDAS;<br>for FILES simple ASCII files are used to communicate,<br>for SOCKETS the communication is established via Unix sockets;<br>defaulted to FILES |

**See also:** SHOW/BACK_MIDAS, WAIT/BACK_MIDAS, DISCONNECT/BACK_MIDAS
SET/BACKGROUND, CLEAR/BACKGROUND, [BackgroundMidas]

**Note:** This command provides a simple way to enforce execution of commands in another Midas session.
If 'method'=FILES, the Midas session we connect to may or may not be in background mode (by executing the command SET/BACKGROUND).
If 'method'=SOCKETS, the Midas session we connect to must already be in background mode.
CONNECT/BACK without parameters will clear any previous "background connections".

**Examples:** `CONNECT/BACK zk nowait z`
Indicate, that all commands with prefix "z," or "Z," will be executed with the prefix stripped off again in background Midas with unit 'zk', do NOT wait until that command terminates.
Z,HELP LOAD/IMAGE would then display the help text in the window belonging to the background Midas with unit 'zk'.

`CONNECT/BACK xk wait b`
Indicate, that all commands with prefix "b," or "B," will be executed with the prefix stripped off again in background Midas with unit 'xk', wait until that command terminates.
B,HELP READ/KEY INPUTC would then display the contents of keyword INPUTC in the window belonging to the background Midas with unit 'xk'.

`CONNECT/BACK xx wait,10 c`
Indicate, that all commands with prefix "c," or "C," will be executed with the prefix stripped off again in background Midas with unit 'xx'.
We wait at most 10 seconds for the termination of the command which has been sent to MIDAS with unit 'xx'. If the command has not finished by then, control returns and an error message is displayed in the current MIDAS.

## CONVERT/TABLE *core* 19-JAN-1994 PB

**Purpose:** Table to image conversion. The values in the input table are used to generate an image in the domain defined by the reference image. Four methods are currently available for this conversion: - (POLYNOMIAL) Polynomial fit to the table data; - (SPLINE) 1D Spline interpolation of the table data. Using the algorithm :Publ. of the Dominion Astrophys. Obs. 204, 1982; - (PLOT) Transform table entries into image pixels. Pixel position is defined by the columns x and y, pixel value is defined by the column z; - (FREQUENCY) The output image is a 2D histogram counting the number of 'events' in the columns x,y of the table

**Syntax:**

```
CONVERT/TABLE image = table x[,y] z refima [method] [par]
```

or

```
CONVERT/TABLE image = table x[,y] refima FREQUENCY
```

| | |
|---|---|
| `image` | output image |
| `table` | input table |
| `x,[y]` | column(s) with the independent variable(s) |
| `z` | column with the dependent variable |
| `refima` | reference image, to define NPIX, START and STEP corresponding to the output image |
| `method` | POLYNOMIAL - polynomial fit; SPLINE - 1D spline interpolation; PLOT - straight table to image transformation; FREQUENCY - 2D histogram distribution; |
| `par` | degree of the polynomial fit (method = POLYNOMIAL, default = 0) or smoothing parameter and degree of the spline (method = SPLINE) |

**See also:** REBIN/TI, INTERPOLATE/TI

**Note:** The method PLOT can be used to display the table on the image display.

The method SPLINE requires that the values in the column used as independent variable are monotonically increasing or decreasing. Only 1D transformations are supported.

Two spline methods are available: – The first one is based on Hermite polynomials and will be used when the smoothing parameter and the degree of the spline are not provided. (Ref: Publ. of the Dominion Astrophys. Obs. 204, 1982). This method doesn't extrapolate , i.e doesn't handle undefined input at the extremes. – The second one is used when the smoothing parameter and the degree of the spline are provided. The smoothing parameter controls the approximation of the interpolated data to the input frame. This parameter has to be choosen carefully : too small s-values will result in an overfitting, too large s-values will produce an underfitting of the data. A typical value to be used would be 1. The degree of the spline is limited to 5.

**Examples:**

```
CONVERT/TABLE OUTPUT = TABLE :X :Y REFIMA SPLINE
```

This command will create the image OUTPUT with the same standard descriptors as REFIMA (the same definition domain), by using a spline interpolation scheme to the points defined in columns :X, :Y in TABLE.

# CONVOLVE/IMAGE

## CONVOLVE/IMAGE *core* 05-NOV-1985 KB

**Purpose:** Convolve image with given point spread function.

**Syntax:** `CONVOLVE/IMAGE frame psf result`

| | |
|---|---|
| `frame` | name of input image |
| `psf` | name of point spread function, number of axes must be not greater than the one of input image |
| `result` | name of result frame |

**See also:** DECONVOLVE/IMAGE, FFT/IMAGE

**Note:** The PSF must be centered at 0.0 (in world coordinates). Please, note, that it is not enough that e.g. a CENTER/GAUSS command confirms that; the actual test in the program (e.g. for x) is:
endx = startx + (xpix-1)*stepx
t = abs(startx) - abs(endx); and the value of t must be close to 0.0
Currently only 1- or 2-dim frames may be processed. This is the same command as DECONVOLVE/IMAGE with no_iter = 0.

**Examples:** `CONVOLVE/IMAGE tijuana juarez matamoros`
Convolve image 'tijuana.bdf' with psf 'juarez.bdf' and store result into newly created image 'matamoros.bdf'

## COPY/DD *core* 17-APR-1991 KB

**Purpose:** Copy descriptors of source frame to destination frame.

**Syntax:** `COPY/DD source_frame source_desc dest_frame dest_desc`

| | |
|---|---|
| `source_frame` | name of source frame |
| `source_desc` | source descriptor |
| `dest_frame` | name of destination frame |
| `dest_desc` | destination descriptor |

**See also:** COPY/LSDD, COPY/KD, COPY/DK

**Note:** The type of the source and destination frame is defaulted to ".bdf" .
COPY/DD in *,1 out copies all descriptors from in.bdf to out.bdf COPY/DD in *,2 out copies all standard descriptors COPY/DD in *,3 out copies all but the standard descriptors COPY/DD in * out is the same as COPY/DD in *,1 out
If destination descriptors do not exist yet, they are created in the dest_frame.

**Examples:** `COPY/DD gallina LHCUTS/R/3/2 gato mystuff/R/12/2`
Will copy elements 3,4 of descr LHCUTS of frame gallina.bdf to elements 12,13 of decscriptor MYSTUFF of frame gato.bdf. Note, that descriptor names are case insensitive.

COPY/DIMA *core* 09-APR-1992 KB

**Purpose:** Copy numerical descriptor of source frame to new image.

**Syntax:** `COPY/DIMA source_frame source_desc dest_frame`

`source_frame` name of source frame

`source_desc` source descriptor, it must be of non_character type

`dest_frame` name of output image

**See also:** COPY/ID, COPY/DK, COPY/KD

**Note:** The result image will always be created with real data type.

**Examples:** `COPY/DIMA tiburon histogram ballena`
Copy the histogram of frame 'tiburon.bdf' to new image 'ballena.bdf'.

`COPY/DIMA tiburon zdata/i/21/100 ballena`
Copy 100 elements of descriptor zdata of frame 'tiburon.bdf' to new image 'ballena.bdf' (starting at the 21st element).

# COPY/DISPLAY

## COPY/DISPLAY *core* 19-SEP-1995 KB

**Purpose:** Make a hardcopy of the ImageDisplay on output_device.

**Syntax:** `COPY/DISPLAY [out_dev] [stop_flg] [ITTdef] [LUTnam] [prflag] [prmode]`

`out_dev` generic MIDAS name or system name of output device, the MIDAS command HELP [PRINTERS] shows a detailed list of all the printers/plotters at your site with their names and locations;
defaulted to LASER, the MIDAS name for the default black+white Postscript Laser printer

`stop_flg` if set to STOP, only read the contents of ImageDisplay and save them in image frame 'screen.ima', but do not create a PostScript file;
defaulted to NOSTOP

`ITTdef` N or P for negative or positive conversion of intensity or I,ITT_name to indicate special ITT to be applied to the data;
defaulted to N for grayscale printers, so black -> white and white -> black;
defaulted to P for colour printers, so the colours on the output will be the same as on the image display.
Note, however, that any ITT already in effect on the display is always applied to the image data. So if you want an exact copy of the screen, do not enter any ITT (just use N or P if you want to exchange black + white in the hardcopy or not).
If you want a different ITT applied to the hardcopy then disable first the ITT on the display, before using COPY/DISPLAY with that specific ITT.

`LUTnam` name of a LUT to be applied to the data; if the loaded image has a real descriptor MIDAS_LUT, this data is taken, if LUT_name is omitted. Else we default to current LUT.

`prflag` print_spec,file_spec
if print_spec = NOPRINT, the PostScript file which is created by COPY/DISPLAY is not sent to 'out_dev';
if print_spec = PRINT, the PostScript file is sent to 'out_dev';
if file_spec = SAMEFILE, the PostScript file is named 'screenXY.ps' where XY is the Midas unit;
if file_spec = NEWFILE, the PostScript file is named 'screenXY:hr:min:sec.ps' where hr:min:sec is the current time;
defaulted to PRINT,SAMEFILE

`prmode` 5-char. flag:
(1) = P(ortrait)/L(andscape), (2) C(olor)/B(lack+white),
(3) = 4/8 bits per pixel on hardcopy,
(4) = N(o background)/B(ackground in blue);
(5) = T(ext)/Z(noText) on output
default is PC8NT for COLOUR, PC8B for SLIDE,
for all other devices the default is PB8NT

**See also:** COPY/ZOOM, COPY/GRAPHICS, ASSIGN/DISPLAY, HELP [Printers]

**Note:** The parameters may also be referenced via:
OUT_DEV= STOP_FLG= ITTDEF= LUTNAM= PRFLAG= PRMODE=
The pixel resolution for the PostScript Laser printers can be controlled by modifying the keyword POSTSCRI(1). Default value is 21.0 (i.e. 21.0*constant pixels per cm). Use HELP/KEYW POSTSCRI to get an explanation of the different elements of POSTSCRI.
An image frame containing the retrieved info is always created, regardless of the stop_flag, its name is the same as the PostScript file with a file type '.ima' instead of '.ps' .
If you work on X11 devices make sure, that the display window is completely inside the screen - otherwise you get an error from the X11 server (at least until release 11.4). Also, make sure that no other window overlaps the display window, else you will find that back on your hardcopy...

**Examples:** `COPY/DISPLAY ps4ipg0 ittdef=p`
Make hardcopy on device 'ps4ipg0' (which happens to be a grayscale Postscript printer at ESO) and keep the same gray levels as in the display window, i.e. no negative conversion of gray levels is done.

`COPY/DISPLAY stop_flag=stop`
Get a copy of display window in Midas image 'screen00.ima' but do not send produce a Postscript file of it (so, also nothing sent to the printer).

`WRITE/KEYW postscri/r/3/3 24,0.1,0.1`
COPY/DISPLAY prmode=pb8nz
Increase the size of the hardcopy a lot and omit the text output at the bottom of the copy.
It depends on the specs of the printer used for the output if and how that works...

`COPY/DISPLAY ? ?  I,lasritt`
Make hardcopy on default device and use the ITT 'lasritt.itt'.

`COPY/DISPLAY PRFLAG=NO,new`
Store "hardcopy" of image display in image 'screen01:11:14:35.ima', convert it to PostScript file 'screen01:11:14:35.ps', but do not print it (assuming that your Midas unit is 01 and the current time is 11:14:35).

---

## COPY/DK *core* 14-MAR-1987 KB

**Purpose:** copy descriptor of source frame to keyword

**Syntax:** `COPY/DK source_frame source_desc dest_key`

| | |
|---|---|
| `source_frame` | name of source frame |
| `source_desc` | source descriptor |
| `dest_key` | name of destination keyword |

**Note:** If the destination keyword does not exist already, it is created with same type and (full) size as the source descriptor.

**See also:** COPY/KD, COPY/DD, COPY/KEYWORD

**Examples:** `COPY/DK tiburon ident/c/10/10 INPUTC`
Will copy the string in descriptor IDENT of frame 'tiburon.bdf' (starting at the 10th element) to keyword INPUTC (starting in the beginning by default).

# COPY/DK

| COPY/DK | *core* | 14-MAR-1987 KB |
|---|---|---|

**Purpose:** copy descriptor of source frame to keyword

**Syntax:** `COPY/DK source_frame source_desc dest_key`

| | |
|---|---|
| `source_frame` | name of source frame |
| `source_desc` | source descriptor |
| `dest_key` | name of destination keyword |

**Note:** If the destination keyword does not exist already, it is created with same type and (full) size as the source descriptor.

**See also:** COPY/KD, COPY/DD, COPY/KEYWORD

**Examples:** `COPY/DK tiburon ident/c/10/10 INPUTC`
Will copy the string in descriptor IDENT of frame 'tiburon.bdf' (starting at the 10th element) to keyword INPUTC (starting in the beginning by default).

## COPY/GRAPHICS *core* 16-Dec-1991 RHW

**Purpose:** Copy the existing plot file to the specified graphic device.

**Subject:** Hardcopy, graphics

**Syntax:** `COPY/GRAPHICS [device] [plotfile]`

`device` plot device; Possible choices can be:
T[ERMINAL] the user graphics terminal (default); in case of a workstation graphics window 0;
D[ISPLAY] the Deanza display or, for workstations, image window 0;
G[WINDOW,n] graphics window n (default 0);
D[WINDOW,n] image window n (default 0);
one of the HARDCOPY devices (see below);
POSTSCRIPT generic postscript device;
NULL the null device; in this case a plot file will be created. Handy if you have no graphic display capabilities availlable.
default is the device that has been assign with ASSIGN/GRAPHICS

`plotfile` MIDAS plotfile (with extension .plt) to be routed. Default is plot currently stored in the system; its name can be found with the command SHOW/GRAPHICS

**Note:** Hardcopy devices should be specified by the system device names. For a number of devices default (MIDAS) names have been implemented. A complete overview of the available devices and their names can be obtained with the command HELP [PRINTERS].
All postscript printers and the device POSTSCRIPT offer the possibility to print in portrait or in landscape mode. In order to get the desired format one has to extend the printer name with ".l" for landscape or ".p" for portrait mode. Default (no extension given) is landscape mode.
The scales of a plot may change if a plot is sent to a plot device other than the original one (pre-specified by ASSIGN/GRAP). In general the axis ratio of the frame will have changed, and hence a square frame WILL NOT BE A SQUARE FRAME ANYMORE if you use COPY/GRAPHICS to dump your plot on another device. This principle also applies in you dump a plot, originally written in portrait mode, in landscape mode, or vice versa.

**See also:** ASSIGN/GRAPHICS, SHOW/GRAPHICS, HELP [PRINTERS]

**Examples:** `COPY/GRAP LASER`
Copy current plot file (SHOW/GRAPH) to the device LASER. The plot will come in landscape orientation (which is the default).

```
ttASSIGN/GRAPHICS ps4ipg1.p NOSPOOLPLOT/TABLE TEST #1 #2 20,10OVERPLOT/TABLE TEST #1
#3COPY/GRAP ps4ipg1.p
```

Assign the laser printer ps4ipg1.p to receive the plot. Here the ESO system device name is used. Plot files will now be created but not be spooled to the device ps4ipg1. Next, plot column 1 versus 2 of table TEST and overplot column 1 versus column 3.
Finally, the plot is sent to ps4ipg1 and will come in portrait mode.

```
ASSIGN/GRAPHICS POSTSCRIPTPLOT/TABLE TEST #1 #2 20,10COPY/GRAPHICS LASER
```

Assign POSTSCRIPT as the output "device". Next, plot column 1 versus column 2 of the table TEST. A file postscript.ps will be created and will contain the graphics. Finally, to make a hardcopy copy, the same plot file to send to the device LASER.

# COPY/ID

## COPY/ID *core* 09-APR-1992 KB

**Purpose:** Copy image data to descriptor of destination frame.

**Syntax:** `COPY/ID source_frame dest_frame dest_desc`

| | |
|---|---|
| `source_frame` | name of source frame |
| `dest_frame` | name of destination frame |
| `dest_desc` | name of destination descriptor |

**See also:** COPY/DIMA, COPY/DK, COPY/KD, COPY/DD

**Note:** The type of the destination frame is defaulted to ".bdf" .
If destination descriptor does not exist yet, it is created in the dest_frame.

**Examples:** `COPY/ID pulpo tiburon imdata/i/1/200`
Copy 200 data values of image 'pulpo.bdf' to the integer descr. 'imdata' of the frame 'tiburon.bdf' - if this descriptor does not yet exist, it will be created.

`COPY/ID pulpo/201/200 tiburon imdata/r/1/200`
Copy 200 data values of image 'pulpo.bdf' (starting at element 201) to the real descriptor 'imdata' of the frame 'tiburon.bdf' - if this descriptor does not yet exist, it will be created.

## COPY/II *core* 28-JUN-1989 KB

**Purpose:** Copy source frame to destination frame.

**Syntax:** `COPY/II source_frame dest_frame dest_format delete_flag`

| | |
|---|---|
| `source_frame` | name of source frame |
| `dest_frame` | name of destination frame |
| `dest_format` | data format of destination frame,<br>I1 for bytes, I2 for 16bit integer, I4 for 32bit integers,<br>R4 for 32bit real, R8 or D for 64bit real (double precision);<br>defaulted to R4 |
| `delete_flag` | if set to 'D', delete input frame after copying; defaulted to 'N' |

**See also:** CREATE/IMAGE

**Note:** The type of the frames is defaulted to ".bdf" .

**Examples:** `COPY/II pollo gallo I1`
Copy image frame pollo.bdf to frame gallo.bdf and convert pixels to 8 bit integer values (bytes).

`COPY/II pollo gallo I1 d`
As above but delete frame pollo.bdf afterwards.

## COPY/IT *core* 02-JUNE-1986 JDP

**Purpose:** Creates a new table file from the input image. The output table contains NPIX(1) rows and NPIX(2) columns of type REAL*4.

**Subject:** Conversion, table

**Syntax:** `COPY/IT inframe outable [column]`

| | |
|---|---|
| `inframe` | input image |
| `outable` | output table |
| `[column]` | for 1-dimensional frames a column name may be given in which the world coordinate of the pixels will be stored. |

**Note:** none

**See also:** COPY/TI

**Examples:** `COPY/IT IMAGE TABLE`

**Restrictions:**
Implemented for 1-D and 2-D frames only.

## COPY/KD *core* 14-MAR-1987 KB

**Purpose:** Copy keyword to descriptor of destination frame.

**Syntax:** `COPY/KD source_key dest_frame dest_desc`

| | |
|---|---|
| `source_key` | name of source kewyord |
| `dest_frame` | name of destination frame |
| `dest_desc` | name of destination descriptor |

**See also:** COPY/LSKD, COPY/DK, COPY/DD

**Note:** The type of the destination frame is defaulted to ".bdf" .
If destination descriptors do not exist yet, they are created in the dest_frame.

**Examples:** `COPY/KD inputc/c/1/10 tiburon ident/c/10/10`
Will copy the first 10 characters of keyword INPUTC to descriptor IDENT of frame tiburon.bdf (starting at the 10th element).

# COPY/KEYWORD

## COPY/KEYWORD *core* 04-JAN-1991 KB

**Purpose:** Copy keywords of same type.

**Syntax:** `COPY/KEYWORD source_key dest_key [M_unit]`

| | |
|---|---|
| `source_key` | keyspecs of source key |
| `dest_key` | keyspecs of destination key, must be of same type as source_key, e.g. both real keywords !! |
| `M_unit` | optional unit of another Midas session, then the contents of the source_key of that session are copied to the dest_key of the current session;<br>the other session must have the same Midas startup directory, i.e. same MID_WORK |

**See also:** READ/KEY, WRITE/KEY, SHOW/KEY

**Note:** See the help for [BackgroundMidas] to get more information about running several Midas sessions in parallel.

**Examples:** `COPY/KEY INPUTI/I/10/4 OUTPUTI/I/3/4`
Copy elements 10,11,12,13 of keyword INPUTI to elements 3,4,5,6 of keyword OUTPUTI.

`COPY/KEY INPUTR OUTPUTR xs`
Copy all elements of keyword INPUTR in the Midas session with Midas unit XS to all elements of keyword OUTPUTR in current session.

## COPY/KI *core* 16-NOV-1992 KB

**Purpose:** Copy keyword to new frame.

**Syntax:** `COPY/KI source_key dest_frame`

| | |
|---|---|
| `source_key` | specfication of numerical source kewyord and no. of elements |
| `dest_frame` | name of destination frame, will be created by MIDAS |

**See also:** COPY/KT, COPY/KD, COPY/KEYWORD

**Note:** The type of the destination frame is defaulted to ".bdf" .
The no. of pixels in 'dest_frame' will be determined from 'source_key', the data values will be converted to "real" data type.

**Examples:** `COPY/KI inputi/i/1/10 perro`
Create image frame 'perro.bdf' with 10 pixels and copy the first 10 elements of integer keyword INPUTI to it.

`COPY/KI outputd gato.ima`
Create image frame 'gato.ima' with 20 pixels (the size of double precision keyword OUTPUTD) and copy all elements of OUPUTD to it.

COPY/KT *core* 12-OCT-1983 JDP

**Purpose:** Copy a keyword into table element

**Syntax:** `COPY/KT keyword table column element`

`Several values from the keyword can be copied into`

`the table row, in position defined by the column(s)`

`and the row number`

| | |
|---|---|
| `keyword` | the input keyword |
| `table` | the output table name |
| `column` | the column reference |
| `element` | the element reference |

**Note:** WARNING: this is a rather slow way of transferring information from a keyword into a table. For copying single elements, it is recommended to use:
table,column,row = keyword
But beware, the keyword element keyword is converted into an ASCII string using the format specified in the SET/FORMAT command before the assignment takes place...!

**See also:** WRITE/TABLE, COPY/TK

**Examples:** `COPY/KT OUTPUTR/R/1/2 TABLE1 :INPUT1 :INPUT2 @20`
– copy the two first values of the keyword OUTPUTR into the column INPUT1 and INPUT2, row number 20 of the table TABLE1
`TABLE1,:INPUT1,@20 = OUTPUTR(2)`
– copy the second value of the keyword OUTPUTR into the column INPUT1, row number 20 of the table TABLE1

# COPY/LSDD

## COPY/LSDD *core* 13-SEP-1995 KB

**Purpose:** Copy descriptors of source frame to descriptors of destination frame; the source and destination descriptors are stored in an ASCII file.

**Syntax:** `COPY/LSDD list source_frame dest_frame`

| | |
|---|---|
| `list` | ASCII file containing a list of source descriptors and corresponding destination descriptors |
| `source_frame` | name of frame with the source descriptors |
| `dest_frame` | name of frame with the destination descriptors |

**Note:** Each record of the ASCII list contains a source descriptor spec. and, separated by at least one blank, a destination descriptor spec.
Descriptors are specified in exactly the same the way as in the command COPY/DK or COPY/KD.
If destination descriptors do not exist yet, they are created in the dest_frame.
If only a source descriptor is specified, the destination descriptor is set to the source descr.

**See also:** COPY/DD, COPY/KD, COPY/DK

**Examples:** `COPY/LSDD tecate.dat bohemia corona`
Copy the contents of the descriptors of bohemia.bdf which are stored in tecate.dat to the descriptors of corona.bdf specified in the same records of tecate.dat.
The records of tecate.dat would look like:
o_time/d/1/8 o_time/d/1/8
ident/c/10/8 new/c/1/8
window_from
. . .

## COPY/LSDK *core* 16-JAN-1992 KB

**Purpose:** Copy descriptors of source frame to keywords; the source descriptors and destination keywords are stored in an ASCII file.

**Syntax:** `COPY/LSDK list source_frame`

| | |
|---|---|
| `list` | ASCII file containing a list of source descriptors and corresponding destination keywords |
| `source_frame` | name of frame with the source descriptors |

**Note:** Each record of the ASCII list contains a source descriptor spec. and, separated by at least one blank, a destination keyword spec.
Descriptors and keywords are specified in exactly the same the way as in the command COPY/DK.
The destination keywords must already exist.

**See also:** COPY/DK, COPY/KD, COPY/LSKD

**Examples:** `COPY/LSDK tecate.dat bohemia`
Copy the contents of the descriptors of bohemia.bdf which are stored in tecate.dat to the keywords specified in the same records of tecate.dat.
The records of tecate.dat would look like:
o_time/d/1/8 inputd/d/2/8
ident/c/10/8 in_a/c/1/8
. . .

## COPY/LSKD *core* 16-JAN-1992 KB

**Purpose:** Copy keywords to descriptors of destination frame; the source keywords and destination descriptors are stored in an ASCII file.

**Syntax:** `COPY/LSKD list dest_frame`

| | |
|---|---|
| `list` | ASCII file containing a list of source keywords and corresponding destination descriptors |
| `dest_frame` | name of frame with the destination descriptors |

**Note:** Each record of the ASCII list contains a source keyword spec. and, separated by at least one blank, a destination descriptor spec.
Descriptors and keywords are specified in exactly the same the way as in the command COPY/KD.

**See also:** COPY/KD, COPY/DK, COPY/LSDK

**Examples:** `COPY/LSKD tecate.dat bohemia`
Copy the contents of the keywords which are stored in tecate.dat to the descriptors of bohemia.bdf which are specified in the same records of tecate.dat.
The records of tecate.dat would look like:
inputd/d/2/8 o_time/d/1/8
in_a/c/1/8 ident/c/10/8
. . .

## COPY/TABLE *core* 12-OCT-1985 JDP

**Purpose:** Copy table files. The output table will consist of the same columns as the input table. The command can be used in connection with the SELECT/TAB command to produce subtables of the input table.

**Syntax:** `COPY/TABLE intable outable [organization]`

| | |
|---|---|
| `intable` | input table name |
| `outable` | output table name |
| `organization` | optional organization of the output file as<br>RECORD (table stored row per row) or<br>TRANSPOSED (table stored columnwise)<br>By default the output table will be organized in the same way than the input table |

**Note:** All the non-standard descriptors from the input table will be copied into the output table. The descriptor HISTORY of the input table will also be copied into the output table.

**See also:** PROJECT/TAB, COPY/TT

**Examples:** `COPY/TABLE TABLE1 OUTPUT`
This will copy 'TABLE1' into output table 'OUTPUT'.

# COPY/TI

## COPY/TI *core* 02-JUNE-1986 JDP

**Purpose:** Creates a new image file from the input table. The command produces a 2D image with NPIX(1) equals to the number of rows and NPIX(2) equals to the number of columns in the table. The columns in the table have to be of the same type, REAL*4 in the current version. The null elements in the table will be set in the image to the "null value" defined by the keyword NULL.

**Syntax:** `COPY/TI intable outimage`

| | |
|---|---|
| `intable` | the input table |
| `outimage` | the output image |

**Note:** none

**See also:** COPY/IT

**Examples:**
```
COPY/TI TABLE IMAGE
```

## COPY/TK *core* 12-OCT-1983 JDP

**Purpose:** Copy one or more table elements into a keyword.

**Syntax:** `COPY/TK table [column ...] element keyword`

| | |
|---|---|
| `table` | the input table name |
| `column` | the column reference |
| `element` | the element reference |
| `keyword` | the output keyword |

**See also:** COPY/KT

**Note:** WARNING: this is a slow way of transferring information from a table into a keyword and only recommended if several columns are involved. For copying single elements, it is recommended to use:
keyword = table,column,row
But beware, the table element table,column,row is converted into an ASCII string using the format specified in the SET/FORMAT command before the assignment takes place...!

**Examples:**
```
COPY/TK vilspa :INPUT1,:INPUT2 @20 outputr/r/1/2
```
Copy the row 20 of the column INPUT1 and INPUT2 from the table 'vilspa.tbl' into the keyword OUTPUTR.
```
outputr(1) = vilspa,:INPUT1,@20
```
Copy the element in row 20 of the column :INPUT1 from the table 'vilspa.tbl' into the keyword OUTPUTR.

COPY/TT *core* 12-OCT-1983 JDP

**Purpose:** Copy a table column to another existing table

**Syntax:** `COPY/TT intable incolumn [outable] outcolumn`

| | |
|---|---|
| `intable` | input table name |
| `incolumn` | input column reference |
| `outable` | optional output table name. Default is the input table The output table MUST already exist. |
| `outcolumn` | output column label |

**Note:** If the reference column is not defined, entries are copied sequentially. If the reference column is defined both in the input and output tables, entries with equal reference value are copied. The reference column for a table is defined with the command SET/REFCOLUMN. The functions are the same if you use the same table as input and output. Note that in this case, if the reference column is not defined (sequence number is acting as reference), then the N selected entries are copied to the first N rows. This command can be used to transform a column of a certain type to a column of an another type.

**See also:** PROJECT/TAB, COPY/TAB, SET/REFC

**Examples:**
```
COPY/TT TABLE1 :INPUT TABLE2 :OUTPUT
```

# COPY/ZOOM

COPY/ZOOM *core* 11-JAN-1994 KB

**Purpose:** Make a copy of the zoom window of the Image Display on output device.

**Syntax:** `COPY/ZOOM [out_dev] [stop_flg] [ITTdef] [LUTnam]`

`[prflag] [prmode]`

| | |
|---|---|
| `out_dev` | generic MIDAS name or system name of output device, the MIDAS command HELP [PRINTERS] shows a detailed list of all the printers/plotters at your site with their names and locations;<br>defaulted to LASER, the MIDAS name for the default black+white Postscript Laser printer |
| `stop_flg` | if set to STOP, only read the contents of zoom window and save them in frame 'screen.ima', but do not send it to an output device; defaulted to NOSTOP |
| `ITTdef` | N or P for negative or positive conversion of intensity or I,ITT_name to indicate special ITT to be applied to the data;<br>defaulted to N, so black -> white and white -> black<br>Note however, that any ITT already in effect on the display is always applied to the image data. So if you want an exact copy of the screen, do not enter any ITT (just use N or P if you want to exchange black + white in the hardcopy or not).<br>If you want a different ITT applied to the hardcopy then disable first the ITT on the display, before using COPY/ZOOM with that specific ITT. |
| `LUTnam` | name of a LUT to be applied to the data, if the loaded image has a real descriptor MIDAS_LUT, this data is taken, if LUT_name is omitted. Else we default to current LUT. |
| `prflag` | if set to NOPRINT, the PostScript file named 'screen.ps' which is created by COPY/ZOOM is not sent to 'out_dev';<br>defaulted to PRINT |
| `prmode` | 5-char. flag:<br>(1) = P(ortrait)/L(andscape), (2) C(olor)/B(lack+white),<br>(3) = 4/8 bits per pixel on hardcopy,<br>(4) = N(o background)/B(ackground in blue);<br>(5) = T(ext)/Z(noText) on output<br>default is PC8NT for COLOUR, PC8B for SLIDE,<br>for all other devices the default is PB8NT |

**See also:** COPY/DISPLAY, CREATE/ZOOM, VIEW/IMAGE, HELP [Printers]

**Note:** This command only works for X11 devices. The zoom window has to exist and be completely inside the display.
Frame screen.ima is always created, regardless of stop_flag.
The command VIEW/IMAGE also has options to make hardcopies of the zoom window.

**Examples:** `COPY/ZOOM ? ? lasritt`
Make hardcopy of zoom window of "active" image display and use the ITT 'lasritt.itt'.

`COPY/ZOOM P5=NOPRINT`
Store "hardcopy" of zoom window of image display only in image frame 'screen.ima' and convert it to PostScript file 'screen.ps'.

## CREATE/ACAT *core* 14-JUL-1994 KB

**Purpose:** Create a catalog of files in the current directory.

**Syntax:** `CREATE/ACAT [cat_name] [dir_spec]`

| | |
|---|---|
| `cat_name` | name of catalog, defaulted to 'acatalog.cat' |
| `dir_spec` | file specifications as used in the VMS '$ DIRECTORY' or UNIX '$ ls' command, defaulted to '*.*';<br>or table,label if the filenames are specified in a column of a Midas table;<br>if set to 'NULL', then the catalog is created without entries |

**See also:** READ/ACAT, DELETE/ACAT, ADD/ACAT

**Note:** The file type of the MIDAS catalog is '.cat' . The catalog contains for each file an entry with a description if it can be derived from the file type, e.g. MIDAS procedure if type = '.prg', else the entry will be: ASCII file.
A file can then be accessed either via its name or via #NO , where NO is its sequence number (if it's in the currently active ASCII file catalog), resp. via cat_name#NO, where NO is its sequence number in the ASCII file catalog cat_name.
Since all files may be in an ASCII file catalog, also images or tables can have an entry there.

**Examples:** `CREATE/ACAT`
Create the ASCII file catalog 'acatalog.cat' with entries for all files in your current directory

`CREATE/ACAT special null`
Create the ASCII file catalog 'special.cat' with initially no entries; entries may be filled via ADD/ACAT commands later on

`CREATE/ACAT juarez matamoros,:files`
Create the ASCII file catalog 'juarez.cat' with entries for all files stored in column :files of table 'matamoros.tbl'.

`create/acat special n*`
Create the ASCII file catalog 'special.cat' with entries for all files beginning with the character 'n'.

# CREATE/COLUMN

## CREATE/COLUMN *core* 12-OCT-1983 JDP

**Purpose:** Creates a new column in a table. Values in the column are initialized to NULL. Information associated with that column is also initialized. A column may have a depth, i.e each element of the table may be itself an array.

**Syntax:** `CREATE/COLUMN table column [unit] [format] [type]`

`table` table name

`column` column reference (by label)

`unit` optional units included in double quotation marks. By default blanks are used as units

`format` format associated with the column according to the FORTRAN-77 rules, with some extensions. The format is used by default when the table is displayed (commands READ/TABLE and PRINT/TABLE). Possible formats are (lower-case edit the sign)
for characters: Aww
for integers: Iww iww (sign edited)
Xww Oww for hexa / octal
Tww.dd tww.dd for Date+Time
(seconds since 1970)
Zww zww zero-filled
for floating: Fww.dd fww.dd (sign edited)
Eww.dd eww.dd (sign edited)
Gww.dd
Rww.dd rww.dd for Right Ascensions
Sww.dd sww.dd for Sexagesimal (decl.)
Tww.dd tww.dd for Date+Time (JD)
Zww.dd zww.dd zero-filled
Default format is E12.6 for R*4,D24.17 for R*8,I11 for I*4

`type` column type including its dimension within parenthesis Possible types are
R*4(array_size) real single precision (default)
R*8(array_size) real double precision
C*n character string, n bytes length
I*n(array_size) signed integer (n = 1 , 2 or 4)
U*n(array_size) unsigned integer ( n = 2 or 4)

Default array_size is 1.

**See also:** NAME/COLUMN, PLOT/TABLE

**Note:** 1) The special function M$EXISTC(table,label) may be used at the command level to check whether a column exists or not. The function returns -2 if the table doesn't exist, the physical number of the column or -1 of the column doesn't exist. 2) Many of the MIDAS commands are not able yet to deal with 3-D tables. If a column is declared as being an array of dimension 12, these commands will only be able to access, for each row of the table, the first element of the array. An exception is the command PLOT/TAB: this command is able to plot the data points in a 3-dim table.

**Examples:**
```
CREATE/COLUMN mytable :RADVEL "KM.SEC-1" E12.3

CREATE/COLUMN mytable :FLUX R*4(12) "Jy" E12.3
```
creates a column FLUX made of 12 real numbers

CREATE/COMMAND *core* 12-OCT-1983 KB

**Purpose:** Create a "user" command in MIDAS.

**Syntax:** `CREATE/COMMAND comnd text`

| | |
|---|---|
| `comnd` | new command with optional qualifier |
| `text` | command line (max. 29 characters) which will be executed when entering "comnd" . Any input following that command will be appended with a preceding blank to the command line given in "text". |

**Note:** As a security measure, you normally cannot create commands which have the same command and qualifier as any existing MIDAS command.
But, if you really want to do it, set your user level to EXPERT with the command SET/MIDAS_SYS - then you can also overwrite the definitions of existing MIDAS commands (be careful...).
Your private login.prg file is a good place to store often used MIDAS command abbreviations and definitions.
All command definitions are cleared after the command BYE.

**See also:** CREATE/D_COMMAND, SET/MIDAS_SYS

**Examples:** `CREATE/COMMAND RK READ/KEY`
Create an abbreviation for the command READ/KEY.

`CREATE/COMMAND XYZ/IMAGE @@ myproc`
Create new command XYZ/IMAGE.
Entering: xyz/ima inframe outframe
will then work like: @@ myproc inframe outframe

# CREATE/CURSOR

CREATE/CURSOR *core* 05-JUL-1995 KB

**Purpose:** Create a cursor window for an existing display window.

**Subject:** Image Display

/sy Syntax: CREATE/CURSOR [dspid] [wind_specs] [Xstation]

| | |
|---|---|
| `dspid` | display identification of main (connected) display window; defaulted to 0 |
| `wind_spec` | xdim,ydim,xoff,yoff of cursor window; defaulted to 180x180 pixels |
| `Xstation` | name of Xworkstation/Xterminal screen (in X11 syntax) where the cursor window should be created; defaulted to the screen where the main display window is |

**See also:** DELETE/CURSOR, CREATE/ZOOM, DELETE/ZOOM, CREATE/DISPLAY, CREATE/GRAPHICS, GET/CURSOR

**Note:** This command is only available on XWindow workstations.
If a cursor window has been created all subsequent cursor readings from the main display window will show in the cursor window a 9x9 area around the cursor #0 (i.e. the mouse) zoomed up by a fctor of 20. This cannot be changed by the values xdim,ydim of the parameter 'wind_spec', just the window size changes!
The display window for which we want to create a cursor window must be currently active!

**Examples:** `CREATE/CURSOR`
Create a cursor window for the curently active display window.

`CREATE/CURSOR 1 ? lw7:0.0`
Create a cursor window for the window with display id = 1 on screen named lw7:0.0 .

## CREATE/D_COMMAND *core* 26-AUG-1985 KB

**Purpose:** Create a directory "user" command

**Syntax:** `CREATE/D_COMMAND comnd text`

| | |
|---|---|
| `comnd` | new command with optional qualifier |
| `text` | line (max. 29 characters) which will be executed when entering "comnd" . All input following that command will be appended without any preceding blank to the command line given in "text". |

**See also:** CREATE/COMMAND

**Note:** All command definitions are cleared after the command BYE.

**Examples:** `CREATE/D_COMMAND EXECUTE/PROC @@ MYDISK:[DIR.SUBDIR]`
creates new user command EXECUTE/PROC, entering:
EXEC/PROC mine
will then yield the translated command:
@@ MYDISK:[DIR.SUBDIR]mine (for VMS !)

`CREATE/D_COMMAND EXECUTE/PROC @@ /mydir/subdir/myprocs/`
creates new user command EXECUTE/PROC, entering:
EXEC/PROC mine
will then yield the translated command:
@@ /mydir/subdir/myprocs/mine (for Unix !)

## CREATE/DEFAULT *core* 12-OCT-1983 KB

**Purpose:** Create special defaults for MIDAS commands.

**Syntax:** `CREATE/DEFAULT comnd def1 def2 ... def8`

| | |
|---|---|
| `comnd` | "command/qualifier" for which default values will be defined |
| `def1` | the default value for parameter 1 |
| | ... |
| `def8` | the default value for parameter 8 |

**See also:** SHOW/DEFAULT, DELETE/DEFAULT

**Note:** The default values indicate via their position for which of the 8 possible parameters of a MIDAS command they are meant, therefore, all preceding "unused" defaults have to be indicated with a question mark (?).

**Examples:** `CREATE/DEFAULT LOAD/IMAGE ? ? 2,2`
Default the scale parameter of the LOAD/IMAGE command to 2,2.

`CREATE/DEFAULT READ/IMAGE ? @20,@20,33`
Default the pixel_specs of the READ/IMAGE command to @20,@20,33.

# CREATE/DISPLAY

## CREATE/DISPLAY *core* 19-Feb-1993 KB

**Purpose:** Create a display window (using XWindow).

**Subject:** Image Display

**Syntax:** `CREATE/DISPLAY [dspid] [dspinfo] [meminfo] [alph_flag] [gsize]`

`[Xstation]`

| | |
|---|---|
| `dspid` | single digit display identification number in [0,9] for a "main" display window, or single character in [a,z] for a shadow display window; defaulted to 0 |
| `dspinfo` | xdim,ydim,xoff,yoff for main display window,<br>size in x,y, offset in x,y (lower left corner is 0,0);<br>or 'shadow,refid' for a "shadow" window, where 'refid' is the id of the display window to be shadowed;<br>defaulted to 512,512,630,330 (i.e. assuming a "main" window) |
| `meminfo` | nomem,xm,ym;<br>no. of image memories (channels), defaulted to 2 for dsp_id 0, to 1 for all others;<br>memory size in x, y - defaulted to size of display window; |
| `alph_flag` | Y(es) or N(o), for using also an alpha_memory or not, defaulted to Y |
| `gsize` | no. of graphic line segments saved;<br>this parameter is important for the refreshing of the window, because only as many line segments can be redrawn as are saved internally;<br>default is 30000 for dsp_id 0, 10000 for all others |
| `Xstation` | name of Xworkstation/Xterminal screen (in X11 syntax) on which the display window should be created,<br>defaulted to 'default' which indicates the local screen |

**See also:** DELETE/DISPLAY, RESET/DISPLAY, INITIALIZE/DISPLAY, CREATE/GRAPHICS
ASSIGN/DISPLAY, COPY/DISPLAY

**Note:** Shadow displays "shadow" (repeat) all operations on the referenced "main" display window. They are useful for showing the contents of your display window elsewhere in the network. There may be more than one shadow display window for a given reference window.
Shadow displays take all input parameters from the referenced "main" display window, except 'Xstation'. Consequently, the parameters 'meminfo', 'alph_flag' and 'gsize' are not used. The display size is taken from the reference display, but the offset of the shadow display can be specified via 'shadow,refid,offx,offy' instead of 'shadow,refid'.
An overlay/graphics memory is always available and will have the highest channel no.
The parameter 'gsize' has no influence on the drawing of lines, it's only important for the refreshing of the window.
When creating a display window, this display becomes the 'active' display which all display related commands will act on.
To navigate among different displays use the command ASSIGN/DISPLAY which will change the currently 'active' display.
You have to have access rights and somebody must be logged in if you want to use a different Xstation than the one you're sitting at.
Keyword DAZDEVR(10) will hold the display id (if "main" display).

**Examples:** `CREATE/DISP 3 400,200,120,320 2`
Create a display (id = 3) of size 400*200 pixels with two image memories, which will be referred to later on as channel 0 and 1, the overlay channel will be channel 2. Also use alphanumerics window and default fonts.

```
CREATE/DISPLAY P5=75000
```
Create a display (id = 0) of size 512*512 pixels with two image memories, which will be referred to later on as channel 0 and 1, the overlay channel will be channel 2. Also use alphanumerics window and default fonts.
We plan to use the overlay channel of the display as a plot device later on, so we reserve space for 75000 line segments to be saved.

```
CREATE/DISPLAY p6=ws4:0.0
```
Create display window on Xstation which has the name 'ws4'.

```
CREATE/DISPLAY a shadow,0 p6=xt65:0.0
```
Create shadow display window of display with id 0 on X11-terminal with name 'xt65'. All operations on display window 0 like e.g. image loading, cursor movements, etc. are repeated on the shadow display.

# CREATE/FCAT

## CREATE/FCAT *core* 14-JUL-1994 KB

**Purpose:** Create a catalog of fit files in current directory..

**Syntax:** `CREATE/FCAT [catname] [dir_spec]`

| | |
|---|---|
| `catname` | name of catalog, defaulted to 'fcatalog.cat' |
| `dir_spec` | file specifications as used in the VMS '$ DIRECTORY' or UNIX '$ ls' command; or table,label if the filenames are specified in a column of a Midas table; if set to 'NULL', the catalog is created without entries; defaulted to '*.fit' |

**Note:** The file type of a MIDAS catalog is '.cat' . The catalog contains contains for each fit file an entry with:
sequence no., name
The catalog 'xyz' is updated automatically by the system, if the catalog is the currently active fit file catalog (use the command "SET/FCAT xyz" to do that). I.e. creating a new fit file will also add a new entry to the catalog, whereas the command ADD/FCAT has to be used explicitly to add an entry to any other fit file catalog.
A fit file can then be accessed either via its name or via #NO , where NO is its sequence number (if it's in the currently active fit file catalog), resp. via #NO_catname, where NO is its sequence number in the fit file catalog catname.
Use the command CREATE/FCAT whenever your catalog does not reflect the actual directory because you executed some operating system commands like '$ COPY' (for VMS) or '$cp ' (for Unix) !

**See also:** READ/FCAT, SET/FCAT, CLEAR/FCAT, SHOW/FCAT, DELETE/FCAT

ADD/FCAT, SUBTRACT/FCAT, SORT/FCAT, SEARCH/FCAT, PRINT/FCAT
CREATE/ICAT, CREATE/TCAT, CREATE/ACAT

**Examples:** `CREATE/FCAT`
Create the fit file catalog 'fcatalog.cat' with entries for all fit files with filetype '.fit' in your current directory

`CREATE/FCAT torreon null`
Create the fit file catalog 'torreon.cat' with initially no entries; it may be filled via ADD/FCAT commands later on

`CREATE/FCAT mazatlan n*.fit`
Create the fit file catalog 'mazatlan.cat' with entries for all fit files matching n*.fit, e.g. n.fit, n123.fit, net.fit ...

`CREATE/FCAT juarez matamoros,:files`
Create the fit catalog 'juarez.cat' with entries for all fit files stored in column :files of table 'matamoros.tbl'.

`create/fcat guadalajara n*.fit,z*`
Create the fit file catalog 'guadalajara.cat' with entries for all fit files matching n*.fit and z* .

## CREATE/FILTER *core* 09-DEC-1985 KB

**Purpose:** Create a filter image (e.g. to be used in the Fourier domain).

**Syntax:** `CREATE/FILTER frame [dim_specs] [frame_specs] [filt_type] [coefs]`

| | |
|---|---|
| `frame` | name of filter frame |
| `dim_specs` | specification of dimensions of filter frame<br>(a) NAXIS, NPIX(1),...,NPIX(NAXIS)<br>(b) "=" (equal sign) to indicate that filter frame shall inherit dimensions from an existing frame;<br>defaulted to 1,512 (i.e. to option (a)) |
| `frame_specs` | more specifications for the filter frame<br>for case (a) this par. has the form:<br>starts,steps = START(1),...,START(NAXIS),STEP(1),...,STEP(NAXIS);<br>for case (b) this par. is:<br>ref_frame = reference frame, take START, STEP and all other standard descriptors from that frame;<br>defaulted to 0.0 for START(i) and 1.0 for STEP(i) |
| `filt_type` | type of filter used:<br>BLPF for Butterworth lowpass filter,<br>ELPF for exponential lowpass filter,<br>BHPF for Butterworth highpass filter,<br>EHPF for exponential highpass filter;<br>filt_type is defaulted to BLPF |
| `coefs` | coefficients for the filter,<br>number and meaning depends on the filt_type |

**Note:** For BLPF, the coefficients represent Do, n, factor; as used in the formula:
f(x,y) = 1. / (1. + factor*(D/Do)**2n)
where D := sqrt( (x-xc)**2 + (y-yc)**2 ) the Euclidean distance from the center coordinates (xc,yc), default for
Do = (xstep+ystep)*(xdim+ydim)/16., n = 1 and factor = 1.
For ELPF the coefficients represent Do, n, factor; as used in the formula:
f(x,y) = exp( -factor*(D/Do)**n); defaults as above.
For BHPF, the coefficients represent Do, n, factor; as used in the formula:
f(x,y) = 1. / (1. + factor*(Do/D)**2n) if: (x,y) .NE. (xc,yc)
f(x,y) = 0. if: (x,y) .EQ. (xc,yc)
defaults as above.
For EHPF the coefficients represent Do, n, factor; as used in the formula:
f(x,y) = exp( -factor*(Do/D)**n) if: (x,y) .NE. (xc,yc)
f(x,y) = 0. if: (x,y) .EQ. (xc,yc)
defaults as above
Use the MIDAS procedure fftfilt.prg in APP_PROC to easily apply these filters to FFT's of images.
For more info use HELP/APPL fftfilt.

**See also:** CREATE/IMAGE, CREATE/RANDOM

**Examples:** `CREATE/FILTER newfilt 2,128,128`
Create Butterworth lowpass filter newfilt.bdf with 128*128 pixels, start values 0.,0. and stepsize 1.,1. (the default values).
Note that NPIX(1) and NPIX(2) are powers of 2, so that the filter may be easily applied to Fourier transforms of images.

# CREATE/GRAPHICS

## CREATE/GRAPHICS *core* 19-Feb-1993 RHW

**Purpose:** Create a graphics window (using XWindow)

**Subject:** Graphics

**Syntax:** `CREATE/GRAPHICS [graph_id] [graph_spec] [gsize] [Xstation]`

| | |
|---|---|
| `graph_id` | single digit identification number in [0,3] for a "main" graphics window, or single character in [a,z] for a shadow graphics window; defaulted to 0 |
| `graph_spec` | xdim,ydim,xoff,yoff, where xdim and ydim are the sizes in x and y, and xoff, yoff the offsets in x and y (lower left corner is 0,0);<br>or 'shadow,refid' for a 'shadow' window, where 'refid' is the id of the graphics window to be shadowed;<br>defaulted to 600,480,0,416 (for DEC-windows 500,480,0,416) |
| `gsize` | no. of graphic line segments saved;<br>this parameter is important for the refreshing of the window, because only as many line segments can be redrawn as are saved internally;<br>defaulted to 100000 |
| `Xstation` | name of Xworkstation/Xterminal screen (in X11 syntax) on which the graphics window should be created,<br>defaulted to 'default' which indicates the local screen |

**See also:** CREATE/DISPLAY, DELETE/GRAPHIC, ASSIGN/GRAPHIC

**Note:** This command is only available on XWindow workstations. Keyword DAZDEVR(11) will hold the graph_id.
Shadow graphics 'shadow' (repeat) all operations on the referenced "main" graphics window. They are useful for showing the contents of your graphics window elsewhere in the network. There may be more than one shadow graphics window for a given reference window.
Shadow graphics take all input parameters from the referenced "main" graphics window, except 'Xstation'. Consequently, the parameter 'gsize' is not used.
The parameter 'gsize' has no influence on the plotting of lines, it's only important for the refreshing of the window.
When creating a graphics window, this window becomes the 'active' graphics window which all graphic related commands will act on.
To navigate among different graphic windows use the command ASSIGN/GRAPHIC which will change the currently 'active' graphics window.
You have to have access rights and somebody must be logged in if you want to use a different Xstation than the one you're sitting at.

**Examples:**

```
CREATE/GRAPHICS 1 400,200,400,400
```

Create a graphics window (id = 1) with an offset of 400 pixels in x,y from the lower left corner and size of 400*200 pixels.

```
CREATE/GRAPHICS P3=120000
```

Create a graphics window (id will be defaulted to 0) with default offset and size.
We plan to plot some very large files and want to make sure, that everything is redrawn when the window is refreshed.

```
CREATE/GRAPHICS p4=ws4:0.0
```

Create graphics window (id will be defaulted to 0) on Xstation which has the name 'ws4'.

```
CREATE/GRAPHICS a shadow,0 p4=xt65:0.0
```

Create shadow graphics window of graphics "main" window 0 on X11-terminal with name 'xt65'. All operations on graphics window 0 like image plotting, cursor movements, etc. are repeated on the shadow graphics window.

CREATE/ICAT *core* 14-JUL-1994 KB

**Purpose:** Create a catalog of images in the current directory.

**Syntax:** `CREATE/ICAT [catname] [dir_spec]`

| | |
|---|---|
| `catname` | name of catalog, defaulted to 'icatalog.cat' |
| `dir_spec` | file specifications as used in the VMS '$ DIRECTORY' or UNIX '$ ls' command; or table,label if the filenames are specified in a column of a Midas table; if set to 'NULL', the catalog is created without entries; defaulted to '*.bdf' |

**Note:** The file type of a MIDAS catalog is '.cat' . The catalog contains for each image frame an entry with:
sequence no., name, contents_of_descriptor IDENT and contents of Naxis, Npix(1), Npix(2).
The catalog 'xyz' is updated automatically by the system, if the catalog is the currently active image catalog (use the command "SET/ICAT xyz" to do that). I.e. creating a new image frame will also add a new entry to the catalog, whereas the command ADD/ICAT has to be used explicitly to add an entry to any other image catalog.
An image can then be accessed either via its name or via #NO , where NO is its sequence number (if it's in the currently active image catalog), resp. via #NO_catname, where NO is its sequence number in the image catalog catname.
Use the command CREATE/ICAT whenever your catalog does not reflect the actual directory because you executed some operating system commands like '$ COPY' (for VMS) or '$cp ' (for Unix) !

**See also:** READ/ICAT, SET/ICAT, CLEAR/ICAT, SHOW/ICAT, DELETE/ICAT
ADD/ICAT, SUBTRACT/ICAT, SORT/ICAT, SEARCH/ICAT, PRINT/ICAT
CREATE/TCAT, CREATE/FCAT, CREATE/ACAT

**Examples:** `CREATE/ICAT`
Create the image catalog 'icatalog.cat' with entries for all image files with filetype '.bdf' in your current directory

`CREATE/ICAT torreon null`
Create the image catalog 'torreon.cat' with initially no entries; entries may be filled via the ADD/ICAT command later on

`CREATE/ICAT mazatlan n*.bdf`
Create the image catalog 'mazatlan.cat' with entries for all image files matching n*.bdf, e.g. n.bdf, n123.bdf net.bdf ...

`CREATE/ICAT juarez matamoros,:files`
Create the image catalog 'juarez.cat' with entries for all image files stored in column :files of table 'matamoros.tbl'.

`create/icat guadalajara n*.bdf,z*`
Create the image catalog 'guadalajara.cat' with entries for all image files matching n*.bdf and z* .

# CREATE/IMAGE

## CREATE/IMAGE *core* 08-JAN-1991 KB

**Purpose:** Create a new image frame.

**Subject:** Artificial frames, models, simulation.

**Syntax:** `CREATE/IMAGE frame [dim_specs] [frame_specs] [func_type] [coefs]`

| | |
|---|---|
| `frame` | name of new image frame |
| `dim_specs` | specification of dimensions of new frame<br>(a) NAXIS, NPIX(1),...,NPIX(NAXIS)<br>(b) "=" (equal sign) to indicate that new frame shall inherit dimensions from an existing frame or interactively via the graphics cursor;<br>defaulted to 1,512 (i.e. to option (a)) |
| `frame_specs` | more specifications for the new frame<br>for case (a) this par. has the form:<br>starts,steps = START(1),...,START(NAXIS),STEP(1),...,STEP(NAXIS);<br>for case (b) this parameter has two possibilities:<br>(b1) ref_frame = reference frame, take START, STEP and all other standard descriptors from that frame;<br>(b2) GCURSOR to indicate interactive input via graphics cursor<br>defaulted to 0.0 for START(i) and 1.0 for STEP(i) |
| `func_type` | function/mode used to calculate the pixel values of the new frame;<br>for option (a) and (b1) the functions are:<br>POLY for polynomial of max. degree 2<br>GAUSS for gaussian function<br>EXPO_DISK for exponential disk function<br>RADIUS_LAW for radius**(1/4) law<br>CIRCLE for frame with centered circle,<br>ELLIPS for frame with centered ellips,<br>ASCII_FILE if data is read in from an ASCII file,<br>TABLE_FILE if data from a table is used,<br>SEQUENCE for linear scale in specified interval,<br>NODATA for a "reference frame" without any data pixel,<br>for option (b2) the functions are:<br>INIT initiates table (default mode, if option(b2))<br>ADD add new points to table<br>DEL delete points from table.<br>Defaulted to POLY. |
| `coefs` | coefficients for the functions listed above, only applicable to options (a) and (b1); number and meaning depends on the parameter func_type;<br>defaulted to 0.0, i.e. complete image will be set to 0. |

**Note:** Using the GCURSOR option (specified as (b1) above), an intermediate table is created with the graphics cursor positions. The positions can be entered in any order.
Note, that modes ADD and DELETE only permit editing of the latest file created with this command!
The following notes apply to options (a) and (b1) only:
for POLY, the coefficients represent a,b,c,d,e,f; up to 6 coeffs. of a polynomial, all missing coeffs. are defaulted to 0.0;
for 1-dim frames: result(x) = a + bx + cx**2;
for 2-dim frames: result (x,y) = a + bx + cy + dxy + ex**2 + fy**2;
for higher dim frames: only result(x,y,...) = a is supported;
for GAUSS the coefficients represent m,s (mean and sigma) for 1-dim. images;
and mx,sx,my,sy (mean, sigma in x, mean, sigma in y) for 2-dim. images;
defaults for mean and sigma are center of image and 3 pixels;
for EXPO_DISK the coefficients represent: amp,scal,pa,i;
the amplitude, the scale length of exponential disk (positive), the position and inclination angle in degrees; pa is calculated from the x-axis, positive is clockwise;
default is 200.,4.,0.,0.
for RADIUS_LAW the coefficients represent amp,er,pa,i;
the amplitude, effective radius, position and inclination angle in degrees, pa is calculated from the x-axis, positive is clockwise;
default is 200.,4.,0.,0.
for CIRCLE, the coefficients represent r,in,out;
radius of circle, value inside and value outside of circle;
for ELLIPS, the coefficients represent a,b,in,out;
the major + minor half-axis of the ellips, value inside and value outside of ellips;
for ASCII_FILE this parameter is the name of the file holding the data in free format;
if the ASCII file holds less pixels than required, the pixels of the newly created image are set to NULL(2), the user defined null value;
for TABLE_FILE this parameter is the name of the table;
this only works for 1-dim images.
The columns :X, :Y are used, the x-values should be non decreasing; intermediate points are calculated via linear interpolation.
for SEQUENCE, the coefficients represent a,b;
specifying an interval [a,b]. The frame will contain a linear scale with first pixel = a and last pixel = b. This option only valid for 1-dim frames; start + step will be 0. and 1.
for NODATA an image frame is created with no data, i.e. this frame consists only of descriptors to be used as a reference frame in MIDAS commands where applicable (the coefficients are irrelevant!)

**See also:** CREATE/FILTER, CREATE/RANDOM, @a func2d

**Examples:** `CREATE/IMAGE new 2,100,100 -200.,-180.,22.,22. POLY 0.,1.`
Create the frame 'new.bdf' with 100*100 pixels, start values = -200.,-180. and stepsize = 22.,22. 'new.bdf' will hold a ramp in x in each line, since it is created as: new(x,y) = 0. + 1.*x .

`CREATE/IMAGE new = sombrero EXPO_DISK`
Create the image 'new.bdf' with same size and standard descriptors as the frame 'sombrero.bdf'. The pixels of 'new.bdf' will have values according to a function generating an exponential disk (with the default values)

`CREATE/IMAGE new = GCURSOR`
Use graphics cursor to create the 1-dim frame 'new.bdf'.

# CREATE/LUT

## CREATE/LUT *core* 08-SEP-1989 KB

**Purpose:** Create a colour lookup table

**Syntax:** `CREATE/LUT LUT_table H_specs S_specs I_specs cyclic_flag`

`CREATE/LUT LUT_table CURSOR [start_LUT] [cursor_LUT]`

| | |
|---|---|
| `LUT_table` | name of new LUT table |
| `H_specs` | start,end,increment for hue, hue in [0.,360.], defaulted to 0.0,360.0,-1.0, |
| `S_specs` | start,end,increment for intensity, intensity in [0,1] defaulted to 0.0,1.0,-1.0 |
| `I_specs` | start,end,increment for saturation, satauration in [0,1] defaulted to 0.0,1.0,-1.0 |
| `cyclic_flag` | CY or NO, for stopping increments at limit or not defaulted to CY |
| `start_LUT` | name of template LUT we begin with, if omitted, currently active LUT is used |
| `cursor_LUT` | name of LUT from which we pull out colours via the cursor |

**Note:** A LUT of 256 levels will be created (as a Midas binary table).
HSI option: if increments < 0., increments will be set to (end-start)/255.
Cursor option: Only works on the IP8000 of Gould-DeAnza! A split screen like with the command SET/SPLIT 1,2 will be set up. If you have an image you want to look at while creating the new LUT it has to be displayed in channel 2.
The start_LUT will be displayed at the bottom of the screen, the cursor_LUT in the upper half. Use the cursor to pick a colour from the upper LUT and process it in the lower LUT in a similar way as with the command MODIFY/LUT .

**Examples:** `CREATE/LUT mycolours 0.,360.,10. 0.8,1.0,-1.`
Use default values for intensity, go from 0. to 360. in hue and 0.8 to 1.0 in saturation to create LUT 'mycolours.lut'.

`CREATE/LUT rojo CURSOR rainbow heat`
Build up new LUT 'rojo.lut' by starting with rainbow and overlaying colours from LUT 'heat.lut' on it.
Only valid for DeAnza displays.

## CREATE/RANDOM_IMAGE *core* 28-FEB-1994 PB

**Purpose:** Create a new image (= bulk data frame + standard descriptors) with pixel values drawn from a random distribution

**Syntax:**

```
CREATE/RANDOM name [dims] [starts,steps] [func_type] [coefs] [seed]
```

```
CREATE/RANDOM name = ref_frame [func_type] [coefs] [seed]
```

| | |
|---|---|
| `name` | name of new frame |
| `ref_frame` | reference frame, i.e. take all descriptors from that frame |
| `dims` | NAXIS, NPIX(1),...,NPIX(NAXIS)<br>default: 2,64,64 |
| `starts,steps` | START(1),...,START(NAXIS),STEP(1),...,STEP(NAXIS)<br>default: = 0. for START(i), = 1. for STEP(i) |
| `func_type` | type of function used for image creation<br>U(NIFORM) for uniform probability function<br>G(AUSS) for Gaussian p.f.<br>E(XPONENTIAL) for exponential p.f.<br>L(OGNORMAL) for lognormal p.f.<br>B(INOMIAL) for binomial p.f.<br>P(OISSON) for Poisson p.f.<br>C(AUCHY) for Cauchy p.f.<br>func_type is defaulted to UNIFORM |
| `coefs` | coefficients for the function above,<br>number and meaning depends on the parameter func_type, defaulted to 0.,1. |
| `seed` | Integer number used as root of the pseudo-random series. Defaulted to the current time in seconds, as provided by M$SECS(). |

**See also:** CREATE/IMAGE

**Note:** for GAUSS the coefficients represent m,s (mean and sigma) for UNIFORM the coefficients represent a,b the endpoints of the interval [a,b].
for POISSON the coefficient represents m the mean

**Examples:**

```
CREATE/RANDOM_IMAGE new 2,100,100 -200.,-180.,22.,22.
```

Will create the frame new.bdf with 100*100 pixels, start values -200.,-180., stepsize 22.,22. the pixel values of new.bdf will be determined according to a uniform distribution over the interval [0,1]

```
CREATE/RANDOM_IMAGE new = sombrero GAUSS 12.5,1.25
```

Will create the image new.bdf with same size and standard descriptors as the frame sombrero.bdf The pixel values s of new.bdf will be determined according to a Gaussian distribution with mean = 12.5 and standard deviation = 1.25

# CREATE/ROW

| CREATE/ROW | *core* | 18-JAN-1993 MP |
|---|---|---|

**Purpose:** add one several rows at a given position of a table

**Syntax:** `CREATE/ROW table row_position number_of_rows`

`table` table name

`row_position` position where the rows have to be added

`number_of_rows`
number of rows to be added

**Note:** The rows will be inserted at the position row_position+1 Rows may also be added in a table using EDIT/TABLE This command is not yet implemented for tables having the RECORD organization

**Examples:** `CREATE/ROW mytable @3 4`
add four rows to the table mytable at the row 3

## CREATE/TABLE *core* 12-AUG-1984 JDP

**Purpose:** Create a table and load values from a file. Format conversion is controlled by an auxiliary format file (.fmt). If the table contains only REAL*4 columns and no NULL values are present, the format file can be omitted.

**Syntax:** `CREATE/TABLE table ncol nrow file [format_file] [organization]`

| | |
|---|---|
| `table` | the table name |
| `ncol` | number of columns; * if ncol defined in format_file. |
| `nrow` | number of rows; * if nrow defined in format_file. |
| `file` | file name, default type .dat with the ASCII info to be loaded into the table. The filename NULL produces an empty table (Disk Space is allocated no columns are created; use CREATE/COL to create them) |
| `format_file` | optional filename (type .fmt) used to define the format of the ASCII file, and the size of the table when ncol and/or nrow are *. By default the name table.fmt is used. If the format file does not exist, the conversion is done via list-directed input, free format. In this case the labels are defaulted to LABxxx. |
| `organization` | optional file organization, as<br>RECORD (table stored row per row) or<br>TRANSPOSED (table stored columnwise) (default). |

**Note:**

The format file describes each field (column) in the table ASCII file as :
DEFINE/FIELD [pos1] [pos2] type [format] label [unit]
where
pos1 = starting position of the field
pos2 = last position in the field
type = data type of the field as I - integer, R - real single precision, D - real double precision, C - character.
format = optional FORTRAN format
label = column label associated to that field
unit = optional units

Three keywords can also be added at the beginning of the format file: ROWS number_of_rows (if not specified in the command line)
COLUMNS number_of_columns (if not specified in the command line)
FS list of field separators used in the ASCII data file. Only used when pos1 and pos2 are not specified in the DEFINE/FIELD statement. Per default tabs and blanks are used as field separators. The list of separators has to be written in the form "f1f2f3". The number of field separators is not limited. If the blank is used as field separator, character strings containing blanks have to be enclosed within double quotes.
Warning: Not all MIDAS commands will work on tables organized in RECORD mode

**See also:** CREATE/COL, CREATE/VIRTUAL, DELETE/COL, WRITE/TAB, EDIT/TAB COMPUTE/TAB SELECT/TAB, COPY/TAB, PROJECT/TAB, COPY/TT, MERGE/TAB

**Examples:**
```
CREATE/TABLE mytable 5 1000 myfile format
```

# CREATE/TABLE

Create the table mytable from ASCII data in the file myfile.dat, using the format in format.fmt
The ASCII file myfile.dat could contain the following structure

```
.........1.........2.........3
12345678901234567890123456789O
NGC 3379 10.75 12.85 E 893
```

The associated file format.fmt is defined as:

```
!  - format file
DEFINE/FIELD 5 8 I :NAME "NGC"
DEFINE/FIELD 10 14 R F5.2 :RA "HOUR"
DEFINE/FIELD 16 20 R F5.2 :DEC "DEGREE"
DEFINE/FIELD 22 22 C :TYPE
DEFINE/FIELD 24 26 I :RV
END
```

```
CREATE/TABLE veron 3 1000 veron veron
```

Generates veron.tbl from the ASCII file veron.dat using the format file veron.fmt which is defined as:

```
FS = ",-"
DEFINE/FIELD R :RA
DEFINE/FIELD R :DEC
DEFINE/FIELD C*13 :NAME
```

No field positions are given, tabs and commas are used as field separators. The file veron.dat has the following structure:

```
0.0047,-35.0596 MS2357-3520
0.0286,-63.5943 MS2357-6352
0.0499,0.0401 Q 2357-0014
0.0528,-0.2427 Q 2357-005A
```

```
CREATE/TABLE table 4 100 data
```

Generates table.tbl from the ASCII file data.dat without associated format file. The information in the file data.dat is in free format (list-directed read in FORTRAN) as

```
1000.0 4.3, 1.1 2.
1001.0 5.2 3.4 0.3
1020.1 2.5 1.2 3.
........
```

CREATE/TCAT *core* 14-JUL-1994 KB

**Purpose:** Create a catalog of tables in the current directory.

**Syntax:** `CREATE/TCAT [catname] [dir_spec]`

| | |
|---|---|
| `catname` | name of catalog, defaulted to 'tcatalog.cat' |
| `dir_spec` | file specifications as used in the VMS '$ DIRECTORY' or UNIX '$ ls' command; or table,label if the filenames are specified in a column of a Midas table; if set to 'NULL', the catalog is created without entries; defaulted to '*.tbl' |

**Note:** The file type of a MIDAS catalog is '.cat' . The catalog contains for each table frame an entry with:
sequence no., name, Nocols, Norows
The catalog 'xyz' is updated automatically by the system, if the catalog is the currently active table catalog (use the command "SET/TCAT xyz" to do that). I.e. creating a new table will also add a new entry to the catalog, whereas the command ADD/TCAT has to be used explicitly to add an entry to any other table catalog.
A table can then be accessed either via its name or via #NO , where NO is its sequence number (if it's in the currently active table catalog), resp. via #NO_catname, where NO is its sequence number in the table catalog catname.
Use the command CREATE/TCAT whenever your catalog does not reflect the actual directory because you executed some operating system commands like '$ COPY' (for VMS) or '$cp ' (for Unix) !

**See also:** READ/TCAT, SET/TCAT, CLEAR/TCAT, SHOW/TCAT, DELETE/TCAT
ADD/TCAT, SUBTRACT/TCAT, SORT/TCAT, SEARCH/TCAT, PRINT/TCAT
CREATE/ICAT, CREATE/FCAT, CREATE/ACAT

**Examples:** `CREATE/TCAT`
Create the table catalog 'tcatalog.cat' with entries for all table files with filetype '.tbl' in your current directory

`CREATE/TCAT torreon null`
Create the table catalog 'torreon.cat' with initially no entries; entries may be filled via the ADD/TCAT command later on

`CREATE/TCAT mazatlan n*.tbl`
Create the table catalog 'mazatlan.cat' with entries for all table files matching n*.tbl, e.g. n.tbl, n123.tbl net.tbl ...

`CREATE/TCAT juarez matamoros,:files`
Create the table catalog 'juarez.cat' with entries for all table files stored in column :files of table 'matamoros.tbl'.

`create/tcat guadalajara n*.tbl,z*`
Create the table catalog 'guadalajara.cat' with entries for all table files matching n*.tbl and z* .

# CREATE/VIRTUAL

## CREATE/VIRTUAL *core* 1-MAR-1994 MP

**Purpose:** Create a virtual table from a physical table. A virtual table doesn't contain any real data except for a selection column. Its other columns are assembled at run-time from the columns of the physical table.

**Syntax:** `CREATE/VIRTUAL virtual table`

| | |
|---|---|
| `virtual` | name of the output virtual table |
| `table` | name of the physical table |

**Note:** No operation modifying the contents of a table is allowed on a virtual table except for the SELECT/TABLE operation. A virtual table may be projected or copied into a physical table. Virtual tables may be used, for instance to access and manipulate local databases, i.e read-only tables without copying these tables into a private directory:

CREATE/VIRTUAL esolv_virtual /midas/calib/data/esolv creates a virtual table esolv_virtual from the physical table /midas/calib/data/esolv SELECT/TABLE esolv_virtual seq.le.1000 selects the first 1000 entries of the virtual table esolv_virtual COPY/TABLE esolv_virtual myesolv copies the selected entries of esolv_virtual into a physical table myesolv

The name of the physical table from which the virtual table is created is stored into the descriptor TVIEWTBL of the virtual table.

**See also:** CREATE/TABLE

**Examples:** `CREATE/VIRTUAL esolv_virtual esolv`
creates the virtual table esolv_virtual.tbl from the table esolv.tbl

CREATE/ZOOM *core* 11-JAN-1994 KB

**Purpose:** Create a zoom window for an existing display window.

**Subject:** Image Display

/sy Syntax: CREATE/ZOOM [dspid] [wind_specs] [Xstation]

| | |
|---|---|
| `dspid` | display identification of main window;<br>defaulted to 0 |
| `wind_spec` | xdim,ydim,xoff,yoff of auxiliary window;<br>defaulted to half of the x- and y- size of main window |
| `Xstation` | name of Xworkstation/Xterminal screen (in X11 syntax) where the auxiliary window should be created;<br>defaulted to the screen where the main display window is |

**See also:** DELETE/ZOOM, COPY/ZOOM
CREATE/DISPLAY, CREATE/GRAPHICS, VIEW/IMAGE, GET/CURSOR

**Note:** This command is only available on XWindow workstations.
The display window for which we want to create a zoom window must be currently active!

**Examples:** `CREATE/ZOOM 1 200,200`
Create a zoom window for the window with display id = 1 and with size of 200*200 pixels.

`CREATE/ZOOM 1 200,200 lw7:0.0`
As above, but create the zoom window on screen named lw7:0.0

`CREATE/ZOOM`
Create a zoom window for display window with id = 0 with size half of x- and y-size of display window 0.

# CUTS/IMAGE

## CUTS/IMAGE *core* 22-MAY-1992 KB

**Purpose:** Display or set low + high cut values of an image frame.

**Syntax:** `CUTS/IMAGE frame [cut_specs]`

| | |
|---|---|
| `frame` | name of image frame |
| `cut_specs` | low,high cuts (a) or reference frame (b) or =option (c)<br>(a) either lowcut,hicut or lowcut or ,hicut to set both cut values or just the low or high value<br>(b) name of data frame the cuts values of which are copied to 'frame'<br>(c) =option, where option can be 'sigma' or 'high'<br>for =sigma, cut values are set to mean-3*sigma, mean+3*sigma<br>for =high, cut values are set to mean-0.1*max, max;<br>if the calculated low (high) cut is less (greater) than min (max) of frame, min (max) is taken instead;<br>if cut_specs are omitted, the contents of the descr. LHCUTS are displayed (i.e. the same as: READ/DESCR frame LHCUTS) |

**See also:** READ/DESCR, LOAD/IMAGE, PLOT/ROW, COPY/DD

**Note:** The cut values will be stored in real descriptor LHCUTS as elements 1,2 and are used with LOAD/IMAGE or PLOT/ROW.
If 'cut_specs' is set to '=sigma', also the values for mean+-sigma and for mean+-2*sigma are computed and stored in real keyword OUTPUTR(11,12,...).
As a reminder:
The descriptor LHCUTS contains also the physical min, max of the image frame in LHCUTS(3,4).

**Examples:** `CUTS/IMAGE durazno 2.2,6.6`
Set low, high cut of durazno.bdf to 2.2 and 6.6 .

`CUTS/IMA durazno pera`
Set low, high cut of durazno.bdf to the cut values of pera.bdf .

`CUTS/IMA durazno =high`
Set low cut value to maximum of (mean-0.1*max) and min of durazno.bdf and set high cut value to max of durazno.bdf .

`CUTS/IMA durazno`
Display current cut values of durazno.bdf .

DEBUG/MODULE *core* 29-MAY-1991 KB

**Purpose:** Run Midas (executable) modules in debug mode.

**Subject:** Debugging

**Syntax:** `DEBUG/MODULE [low_lev,hi_lev] [switch]`

`low_lev,hi_lev`
defines interval of levels, only if the Midas module is executed at a level inside that interval, it is run in debug mode. Modules which are executed at a different level are not (!) run in debug mode.
Defaulted to 1,1.

`switch` ON, NO, TIME or OFF;
ON - run module with debugger;
NO - do not execute the module, i.e. continue with procedure;
TIME - run module normally + show time before and after execution
OFF - disable debug mode, execute module normally;
defaulted to ON

**See also:** DEBUG/PROCEDURE

**Note:** 'Levels' are the levels at which the relevant Midas procedures containing the module are executed.
To turn off debug mode at all levels, enter simply DEBUG/MODU OFF .
If switch = TIME, the times are also stored in the MIDAS logfile.

**Examples:** `DEBUG/MODU 1,2`
Run all Midas modules which are executed in a procedure at level 1 or level 2 in debug mode.

`DEBUG/MODU off`
Turn debug mode off for all levels.

`DEBUG/MODU 2,2 no`
Skip all executable modules at procedure level 2.

# DEBUG/PROCEDURE

## DEBUG/PROCEDURE *core* 17-DEC-1990 KB

**Purpose:** Run Midas procedures in debug mode.

**Subject:** Debugging

**Syntax:** `DEBUG/PROCEDURE [low_lev,hi_lev] [switch]`

`low_lev,hi_lev`
defines interval of levels, only if the Midas procedure is executed at a level inside that interval, it is run in debug mode. Defaulted to 1,1.

`switch` ON or OFF, enable or disable debug mode at given levels; defaulted to ON

**See also:** DEBUG/MODULE, ECHO/ON, TRANSLATE/SHOW

**Note:** In debug mode each command line in the Midas procedure is displayed and only executed after the user hits the RETURN key. Thus you can step through the procedure line by line.
Once you are in the debug mode (indicated by the prompt Mdb> ), enter 'h' for help to get a list of all the available options. E.g. at each step keywords may be inspected.
To turn off debug mode at all levels, enter simply DEBUG/PROC OFF .

**Examples:** `DEBUG/PROC 1,2`
All Midas procedures at level 1 or level 2 are executed in debug mode.

`DEBUG/PROC off`
Turn debug mode off for all levels.

## DECONVOLVE/IMAGE *core* 02-OCT-1985 KB

**Purpose:** Deconvolve image with given point spread function using an iterative algorithm published by L.B.Lucy in the Astronomical Journal, vol. 79., 1974.

**Syntax:** `DECONVOLVE/IMAGE frame psf result [no_iter] [cont_flag]`

| | |
|---|---|
| `frame` | name of input image |
| `psf` | name of point spread function, number of axes must be not greater than the one of input image |
| `result` | name of result frame |
| `no_iter` | number of iterations; defaulted to 3 |
| `cont_flag` | C for continuing iteration; defaulted to NO |

**See also:** CONVOLVE/IMAGE

**Note:** The PSF must be centered at 0.0 (in world coordinates). Please, note, that it is not enough that e.g. a CENTER/GAUSS command confirms that; the actual test in the program (e.g. for x) is:
endx = startx + (xpix-1)*stepx
t = abs(startx) - abs(endx); and the value of t must be close to 0.0
Currently only 1- or 2-dim frames may be processed.
Setting no_iter = 0, will result in the convolution of the input image with the point spread function!
If cont_flag = C, it is assumed, that the output frame holds already the results of a previous iteration.

**Examples:** `DECONVOLVE/IMAGE myframe mypsf result`
Deconvolve image 'myframe.bdf' with psf 'mypsf.bdf', use 3 iterations and store result into newly created image 'result.bdf'.

`DECONVOLVE/IMAGE myframe mypsf result ? C`
As above, but use the values of frame 'result.bdf' as start for iteration.

`DECONVOLVE/IMAGE a psf b 0`
Convolve image 'a.bdf' with point spread function 'psf.bdf' and store resulting image into frame 'b.bdf'.

# DELETE/ACAT

## DELETE/ACAT *core* 14-JAN-1991 KB

**Purpose:** Delete files (on disk) with an entry in given ASCII file catalog.

**Syntax:** `DELETE/ACAT [catalog] [conf_flag] [range]`

| | |
|---|---|
| `catalog` | name of catalog, defaulted to currently active ASCII file catalog (cf. SET/ACAT command) |
| `conf_flag` | CONF or NOCONF for confirming the deletion of each file involved, defaulted to CONF |
| `range` | low,hi indicating the range of catalog entries in which you want to delete frames, defaulted to 1,999999 |

**Note:** Warning: This command does not delete the catalog file itself, instead all the files related to this catalog (i.e. those files having an entry in the catalog)!!!
To delete just the catalog file use the relevant host command ($DELETE (VMS) or $rm (Unix)).
If you decide not to delete a frame (conf_flag = CONF), then the corresponding entry of that frame is not removed from the catalog.
For confirmation answer y (for yes) or anything (for no).

**See also:** SET/ACAT, CLEAR/ACAT, SUBTRACT/ACAT, DELETE/ICAT

**Examples:** `DELETE/ACAT junk ? 12,100`
Delete all files with entries in the ASCII file catalog junk.cat but only those with entry numbers in the interval [12,100]; before each file is deleted, the user has to confirm it.

## DELETE/COLUMN *core* 02-NOV-1985 JDP

**Purpose:** Delete table column(s)

**Syntax:** `DELETE/COLUMN table column_sel`

| | |
|---|---|
| `table` | table name |
| `column_sel` | references, separated by blanks, to the column(s) to be deleted |

**Note:** Column numbers or labels may be used. Use SHOW/TABLE to see the resulting table layout.

**See also:** CREATE/COLUMN

**Examples:** `DELETE/COLUMN mytable #2 :X`
This command deletes the second column and the column :X from mytable

`DELETE/COLUMN mytable :ra :dec`
This command deletes the columns :ra and :dec from mytable

## DELETE/COMMAND *core* 26-AUG-1985 KB

**Purpose:** Delete a user defined command.

**Syntax:** `DELETE/COMMAND [comnd]`

| | |
|---|---|
| `comnd` | command which was defined before by the user<br>if no command is given, all user defined commands are cleared (including the commands enabled via SET/CONTEXT) |

**See also:** SHOW/COMMANDS, CREATE/COMMAND, CREATE/D_COMMAND

**Note:** none

**Examples:** `DELETE/COMMAND APPLY/ALGORITHM`
delete the command APPLY/ALGORITHM

`DELETE/COMMAND`
Delete all dynamically added commands, i.e the ones created by the user and the ones created via SET/CONTEXT commands.

# DELETE/CURSOR

## DELETE/CURSOR *core* 11-JAN-1994 KB

**Purpose:** Delete cursor window(s) on XWindow displays.

**Syntax:** `DELETE/CURSOR [dispno]`

| | |
|---|---|
| `dispno` | no. of display window to which the cursor window belongs<br>or * to indicate all cursor windows;<br>defaulted to * |

**See also:** CREATE/CURSOR, CREATE/DISPLAY, DELETE/DISPLAY, DELETE/ZOOM
Chapter 6 of the MIDAS Users Manual, Volume A

**Note:** The display window 'dispno' must be the currently active display window!
This command is only valid for XWindow Image Displays.

**Examples:** `DELETE/CURSOR`
Delete all currently existing cursor windows.

`DELETE/CURSOR 4`
Delete cursor window connected to the currently active display window with id = 4.

## DELETE/DEFAULTS *core* 12-OCT-1983 KB

**Purpose:** Delete special defaults for command.

**Syntax:** `DELETE/DEFAULTS [comnd]`

| | |
|---|---|
| `comnd` | command/qualifier for which default values should be reset to general MIDAS defaults;<br>if comnd is omitted, all default definitions are cleared;<br>this is the default |

**See also:** CREATE/DEFAULTS

**Note:** none

**Examples:** `DELETE/DEFAULT LOAD/IMAGE`
Remove the special default values which were set before via the command CREATE/DEFAULT for the command LOAD/IMAGE.

## DELETE/DESCRIPTOR *core* 24-OCT-1991 KB

**Purpose:** Delete descriptor of frame.

**Syntax:** `DELETE/DESCR frame descr`

| | |
|---|---|
| `frame` | name of frame |
| `descr` | name of descriptor to be deleted |

**See also:** WRITE/DESCR, READ/DESCR, PRINT/DESCR, SHOW/DESCR

**Note:** If you set descr to '*' (wild card), all descriptors will be deleted.

**Examples:** `DELETE/DESCR caballo instrument`
Remove descriptor instrument from frame caballo.bdf .

`DELETE/DESCR asno *`
Remove all descriptors from frame asno.bdf .

## DELETE/DISPLAY *core* 11-JUN-1991 KB

**Purpose:** Delete display window(s) on XWindow displays.

**Syntax:** `DELETE/DISPLAY [disp]`

| | |
|---|---|
| `disp` | display specification,<br>can be a number (the display_id used in "create/display");<br>or * to indicate all display windows;<br>or ALL to delete all windows, i.e. also the graphic windows;<br>defaulted to * |

**See also:** DELETE/GRAPHICS, CREATE/DISPLAY, CREATE/GRAPHICS
Chapter 06 of MIDAS Users Manual, Volume A

**Note:** With the parameter set to ALL, DELETE/DISPLAY is equivalent to DELETE/GRAPHICS.
This command is only valid for XWindow Image Displays.

**Examples:** `DELETE/DISPLAY`
Delete all display windows including the zoom_windows. Do not delete any graphics window.

`DELETE/DISPLAY 4`
Delete display window with display id 4, if it has a zoom_window connected, also that one is deleted.

`DELETE/DISPLAY all`
Remove all display and graphics windows, which have been previously created by Midas, from the screen.

# DELETE/FCAT

## DELETE/FCAT *core* 14-JAN-1991 KB

**Purpose:** Delete fit files (on disk) with an entry in given catalog.

**Syntax:** `DELETE/FCAT [catalog] [conf_flag] [range]`

| | |
|---|---|
| `catalog` | name of catalog, defaulted to currently active fit_file catalog (cf. SET/FCAT command) |
| `conf_flag` | CONF or NOCONF for confirming the deletion of each file involved, defaulted to CONF |
| `range` | low,hi indicating the range of catalog entries in which you want to delete frames, defaulted to 1,999999 |

**Note:** Warning: This command does not delete the catalog file itself, instead all the fit files related to this catalog (i.e. those files having an entry in the catalog)!!!
To delete just the catalog file use the relevant host command ($DELETE (VMS) or $rm (Unix)).
If you decide not to delete a fit_file (conf_flag = CONF), then the corresponding entry of that fit_file is not removed from the catalog.

**See also:** SET/FCAT, SUBTRACT/FCAT, DELETE/FIT

**Examples:** `DELETE/FCAT spectuv NO 12,100`
delete all files with entries in the fit file catalog spectuv.cat, but only those with entry numbers in the interval [12,100]; no confirmation is required.

## DELETE/FIT *core* 15-JAN-1991 KB

**Purpose:** Delete a fit file.

**Syntax:** `DELETE/FIT name [conf_flag]`

| | |
|---|---|
| `name` | name of fit file, this file (in UNIX) or all versions of it (in VMS) are deleted (on disk) |
| `conf_flag` | CONF or NO, if you want (or do not want) to confirm the deletion of each file, defaulted to CONF |

**Note:** You should rather use this command instead of the relevant host system command, since it preserves the integrity of the active (enabled) fit file catalog.
I.e. if the deleted fit file has an entry in the currently active fit file catalog, its entry in the catalog is also removed.

**See also:** SET/FCAT, DELETE/FCAT

**Examples:** `DELETE/FIT gauss`
Unix: delete file gauss.fit (with confirmation)
VMS: delete all files gauss.fit.* (with confirmation)

`DELETE/FIT poly NO`
delete file poly.fit without confirmation, if poly.fit has an entry in the currently active fit file catalog, this entry is removed from the catalog

## DELETE/GRAPHICS *core* 11-JUN-1991 KB

**Purpose:** Delete graphic window(s) on XWindow displays.

**Syntax:** `DELETE/GRAPHICS [grap]`

| | |
|---|---|
| `grap` | graphics window specification,<br>can be a number, the display_id used in "create/graphics";<br>or * to indicate all graphics windows;<br>or ALL to delete all windows, i.e. also the display windows;<br>defaulted to * |

**See also:** DELETE/DISPLAY, CREATE/DISPLAY, CREATE/GRAPHICS
Chapter 06 of MIDAS Users Manual, Volume A

**Note:** With the parameter set to ALL, DELETE/GRAPHICS is equivalent to DELETE/DISPLAY.
This command is only valid for XWindow Image Displays.

**Examples:** `DELETE/GRAPHICS`
Delete all graphics windows, leave any display window untouched.

`DELETE/GRAPHICS 2`
Delete graphics window with id = 2.

`DELETE/GRAPHICS all`
Remove all graphics and display windows, which have been previously created by Midas, from the screen.

## DELETE/ICAT *core* 14-JAN-1991 KB

**Purpose:** Delete image frames (on disk) with an entry in given catalog.

**Syntax:** `DELETE/ICAT [catalog] [conf_flag] [range]`

| | |
|---|---|
| `catalog` | name of catalog, defaulted to currently active image catalog (cf. SET/ICAT command) |
| `conf_flag` | CONF or NOCONF for confirming the deletion of each file involved, defaulted to CONF |
| `range` | low,hi indicating the range of catalog entries in which you want to delete frames, defaulted to 1,999999 |

**Note:** Warning: This command does not delete the catalog file itself, instead all the images related to this catalog (i.e. those images having an entry in the catalog)!!!
To delete just the catalog file use the relevant host command ($DELETE (VMS) or $rm (Unix)).
If you decide not to delete a frame (conf_flag = CONF), then the corresponding entry of that frame is not removed from the catalog.
For confirmation answer y (for yes) or anything (for no).

**See also:** SET/ICAT, SUBTRACT/ICAT, DELETE/IMAGE

**Examples:** `DELETE/ICAT spectuv ? 12,100`
Delete all frames with entries in the image catalog spectuv.cat, but only those with entry numbers in the interval [12,100]; before each file is deleted, the user has to confirm it.

# DELETE/IMAGE

## DELETE/IMAGE *core* 15-JAN-1991 KB

**Purpose:** Delete an image frame.

**Syntax:** `DELETE/IMAGE name [conf_flag]`

| | |
|---|---|
| `name` | name of image frame, this file (in UNIX) or all versions of it (in VMS) are deleted (on disk) |
| `conf_flag` | CONF or NO, if you want (or do not want) to confirm the deletion of each file, defaulted to CONF |

**Note:** You should rather use this command instead of the relevant host system command, since it preserves the integrity of the active (enabled) image catalog.
I.e. if the deleted image has an entry in the currently active image catalog, its entry in the image catalog is also removed.

**See also:** SET/ICAT, DELETE/ICAT

**Examples:** `DELETE/IMAGE ngc1234`
Unix: delete file ngc1234.bdf (with confirmation)
VMS: delete all files ngc1234.bdf.* (with confirmation)

`DELETE/IMA sombero NO`
delete file sombrero.bdf without confirmation if sombrero.bdf has an entry in the currently active image catalog, this entry is removed from the catalog

## DELETE/KEYWORD *core* 14-JAN-1991 KB

**Purpose:** Delete user defined keyword

**Syntax:** `DELETE/KEYWORD key`

| | |
|---|---|
| `key` | name of keyword, has to be defined before by the user, you cannot (!) delete a keyword defined by the system, i.e. the keywords initially stored in the file MID_MONIT:syskeys.dat.<br>If the keyword name is omitted, no prompting is done - just an error message displayed. |

**Note:** In general it is not a good idea to first create a new keyword and then delete it later on. Inside procedures you should always use DEFINE/LOCAL instead because local keywords are automatically removed upon termination of the procedure!

**See also:** WRITE/KEY, DEFINE/LOCAL_KEY

**Examples:** `DELE/KEY pluto`
Assuming that you entered e.g. "WRITE/KEY pluto/i/1/3 1,2,3 " previously, this will remove the keyword pluto again.

## DELETE/LOGFILE *core* 12-OCT-1983 KB

**Purpose:** Delete current logfile + open a new logfile.

**Syntax:** `DELETE/LOGFILE`

**See also:** PRINT/LOGFILE

**Note:** When starting up Midas the old logfile(s) are automatically deleted.

**Examples:** `DELE/LOG`
Delete current logfile and start with a new one.

## DELETE/ROW *core* 18-JAN-1993 MP

**Purpose:** delete one or several rows of a table

**Syntax:** `DELETE/ROW table row_sel`

| | |
|---|---|
| `table` | table name |
| `row_sel` | sequence numbers of the row(s) to be deleted, commas can be used for enumeration, and a double dot for ranges |

**Note:** Rows may also be deleted in a table using EDIT/TABLE This command is not yet implemented for tables having the RECORD organization

**Examples:** `DELETE/ROW mytable @1..5,8..14`
delete rows 1to 5 and 8 to 14 of the table mytable

## DELETE/TABLE *core* 15-JAN-1991 KB

**Purpose:** Delete an table file.

**Syntax:** `DELETE/TABLE name [conf_flag]`

| | |
|---|---|
| `name` | name of table file, this file (in UNIX) or all versions of it (in VMS) are deleted (on disk) |
| `conf_flag` | CONF or NO, if you want (or do not want) to confirm the deletion of each file, defaulted to CONF |

**Note:** You should rather use this command instead of the relevant host system command, since it preserves the integrity of the active (enabled) table catalog.
I.e. if the deleted table has an entry in the currently active table catalog, its entry in the table catalog is also removed.

**See also:** CREATE/TABLE, SET/TCAT, DELETE/TCAT

**Examples:** `DELETE/TABLE ugv`
Unix: delete file ugv.tbl (with confirmation)
VMS: delete all files ugv.tbl.* (with confirmation)

`DELETE/TAB ugv NO`
Delete file ugv.tbl without confirmation, if ugv.tbl has an entry in the currently active table catalog, this entry is removed from the catalog.

# DELETE/TCAT

## DELETE/TCAT *core* 14-JAN-1991 KB

**Purpose:** Delete table files (on disk) with an entry in given catalog.

**Syntax:** `DELETE/TCAT [catalog] [conf_flag] [range]`

| | |
|---|---|
| `catalog` | name of catalog, defaulted to currently active table catalog (cf. SET/TCAT command) |
| `conf_flag` | CONF or NOCONF for confirming the deletion of each file involved, defaulted to CONF |
| `range` | low,hi indicating the range of catalog entries in which you want to delete frames, defaulted to 1,999999 |

**Note:** Warning: This command does not delete the catalog file itself, instead all the tables related to this catalog (i.e. those tables having an entry in the catalog)!!!
To delete just the catalog file use the relevant host command ($DELETE (VMS) or $rm (Unix)).
If you decide not to delete a table (conf_flag = CONF), then the corresponding entry of that table is not removed from the catalog.

**See also:** SET/TCAT, SUBTRACT/TCAT, DELETE/TABLE

**Examples:** `DELETE/TCAT orders NO 12,100`
delete all files with entries in the table catalog orders.cat, but only those with entry numbers in the interval [12,100]; no confirmation is required.

## DELETE/TEMP *core* 14-JAN-1991 KB

**Purpose:** Delete temporary MIDAS frames.

**Syntax:** `DELETE/TEMP`

See also: DELETE/IMAGE

**Note:** All frames with names of the form &a, &b, ..., &z are used as temporary work frames in MIDAS and their names expanded to middumma.bdf, middummb.bdf, ..., middummz.bdf internally.
These files are deleted with this command.

**Examples:** `DELETE/TEMP`
Delete all middumm* frames in current directory.

## DELETE/ZOOM *core* 11-JAN-1994 KB

**Purpose:** Delete zoom window(s) on XWindow displays.

**Syntax:** `DELETE/ZOOM [dispno]`

| | |
|---|---|
| `dispno` | no. of display window to which the zoom window belongs<br>or * to indicate all zoom windows;<br>defaulted to * |

**See also:** CREATE/ZOOM, CREATE/DISPLAY, DELETE/DISPLAY, DELETE/CURSOR
Chapter 6 of the MIDAS Users Manual, Volume A

**Note:** The display window 'dispno' must be the currently active display window!
This command is only valid for XWindow Image Displays.

**Examples:** `DELETE/ZOOM`
Delete all currently existing zoom windows.

`DELETE/ZOOM 4`
Delete zoom window connected to the currently active display window with id = 4.

## DISCONNECT/BACK_MIDAS *core* 04-AUG-1994 KB

**Purpose:** Disconnect from a previously connected background MIDAS.

**Syntax:** `DISCONNECT/BACK_MIDAS unit`

| | |
|---|---|
| `unit` | unit of background Midas; |

**See also:** CONNECT/BACK_MIDAS, SET/BACKGROUND, CLEAR/BACKGROUND
[BackgroundMidas]

**Note:** This command cuts the connection from the current Midas to another Midas session, i.e. you cannot send commands to it anymore.

**Examples:** `DISCONNECT/BACK zk`
Cut the connection to Midas session with unit ZK. It is assumed that a command: CONNECT/BACK_MIDAS zk has been executed before.

# DISPLAY/CHANNEL

## DISPLAY/CHANNEL *core* 14-APR-1986 KB

**Purpose:** Display contents loaded in an Image Display channel.

**Syntax:** `DISPLAY/CHANNEL [chanl] [LUT_sect]`

| | |
|---|---|
| `chanl` | Image Display channel (or image memory); defaulted to the currently active channel |
| `LUT_sect` | section of LUT to be used, default = 0; this paramater is only applicable for DeAnza displays (!) |

**See also:** CLEAR/CHANNEL, SHOW/CHANNEL, LOAD/IMAGE, ASSIGN/DISPLAY

**Note:** For DeAnza displays the channel may either be specified by it's number (e.g 0) or by the colour it represents in RGB mode, so, R(ed)=channel 0, G(reen)=1, B(lue)=2.
Also for DeAnzas "DISPLAY/CHANNEL ALL" is a special option in RGB_mode to view the R+G+B channels together.
The O(verlay) channel is usually the highest numbered channel.

**Examples:** `DISPLAY/CHANNEL 1`
Display contents of image channel 1.

## DISPLAY/LUT *core* 01-NOV-1985 KB

**Purpose:** En/disable display of currently active colour lookup table at bottom of screen.

**Syntax:** `DISPLAY/LUT [switch] [intens]`

| | |
|---|---|
| `switch` | ON or OFF, defaulted to ON |
| `intens` | intensity for frame around colour bar, value in [0,255], defaulted to 200, also the strings DARK or LIGHT may be given as intensity; this parameter is only applicable for DeAnza displays (!) |

**See also:** LOAD/LUT, LOAD/ITT, MODIFY/LUT

**Note:** For DeAnza image displays, a frame with the given intensity is drawn around the color bar (make sure, the overlay is enabled).

**Examples:** `DISP/LUT`
display color bar and if a DeAnza display, frame it in white

`DISP/LUT OFF`
disable display of LUT

`DISPLAY/LUT on dark`
display the LUT and if a DeAnza display, frame it in black (this is interesting if you have a very light LUT)

## DRAW/ANY *core* 25-JAN-1991 KB

**Purpose:** Draw manually in the overlay channel.

**Syntax:** `DRAW/ANY [intens]`

| | |
|---|---|
| `intens` | intensity (colour) for line drawing,;<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255<br>for X11 displays it's really a colour (see Note) and the par. is defaulted to WHITE |

**See also:** DRAW/RECTANGLE, /CIRCLE, /ELLIPS, /SLIT, /ARROW, /CROSS, /LINE

**Note:** The following colours are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Magenta(7), Cyan(8).
For DeAnza displays, the cursor box has to be set up with cursor-1 on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of a line from last point to current one. To interrupt drawing and start at new point, press ENTER with both cursors switched off. To get out, switch both cursors off and press ENTER twice.
Setting TRACK on is the same as pressing ENTER continuously, so you can achieve very smooth lines like that.
For X11 displays the cursor is moved with the mouse and Use the ENTER button on the mouse to draw. Use the EXIT button to interrupt drawing and start at new point, press the EXIT button twice to really exit.

**Examples:** `DRAW/ANY 5`
draw a graph with intensity=5 (DeAnza) or colour=blue (X11)

# DRAW/ARROW

DRAW/ARROW *core* 22-JUL-1991 KB

**Purpose:** Draw arrows in the overlay channel.

**Syntax:** `DRAW/ARROW [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

`in_spec` input specification for drawing position;
(a) CURSOR if coordinates are chosen via the cursor rectangle;
(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND;
(c) x1,y1,x2,y2 defining the coords. of start and end of arrow
(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 arrows, see parameter 'key_flag'
defaulted to CURSOR

`coord_ref` F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;
defaulted to S, also any "strange" input is interpreted as "S" !

`draw_opt` direction of arrow, only used, if cursor rectangle input:
RU for left_down to right_up, RD for left_up to right_down, LU for right_down to left_up, LD for right_up to left_down in cursor rectangle;
defaulted to RU, also any "strange" input is interpreted as "RU" !

`intens` intensity(color),rotation angle (in degrees) of drawing;
for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255,0.
for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE,0.
the arrow-graph is first built up from the specs and in the very end rotated counter-clockwise with the rotation angle

`nocurs` no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER for the start- and endpoint of each arrow - overrides the direction given in 'draw_opt';
defaulted to 2

`key_flag` K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the positions for up to 10 arrows, the last coord. must be followed by a -1;
defaulted to N

**See also:** DRAW/LINE, DRAW/CIRCLE, DRAW/ELLIPS, DRAW/RECTANGLE, DRAW/SLIT DRAW/CROSS, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of an arrow. To get out, switch both cursors off and press ENTER.
For X11 displays the cursor rectangle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw or to exit.
Due to the redrawing of the cursor rectangle the endpoints of the arrow will not have immediately the desired color, but that will be corrected, once you exit from the command.

**Examples:** `DRAW/ARROW`
use cursor rectangle to define endpoints and draw a white arrow pointing to the upper right corner.

`DRAW/ARROW coords F ? 100,45.`
use values from columns :xstart, :ystart, :xend, :yend of table file coords.tbl, interpret them as real world coordinates of the currently displayed frame and draw arrows with intensity 100 for DeAnza (i.e. gray) or white color for X11; finally rotate the arrows around their center by 45.0 degrees

`DRAW/ARROW 200,200,300,300`
draw an arrow from the screen pixels (200,200) to (300,300)

# DRAW/CIRCLE

DRAW/CIRCLE *core* 05-SEP-1995 KB

**Purpose:** Draw circles in the overlay channel.

**Syntax:** `DRAW/CIRCLE [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

`in_spec` input specification for lower left and upper right corner of the circle;
(a) CURSOR if circle is chosen via the cursor(s);
(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND defining the coords. of lower left and upper right corner of a rectangle surrounding the required circle;
(c) x1,y1,x2,y2 defining the coords. of lower left and upper right corner of a rectangle surrounding the required circle;
(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 circles, see parameter 'key_flag'
defaulted to CURSOR

`coord_ref` F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;
defaulted to S, also any "strange" input is interpreted as "S"

`draw_opt` F(ill) or N(ofill) the circle;
defaulted to N, also any "strange" input is interpreted as "N"

`intens` intensity(color) of drawing;
for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255.
for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE.

`nocurs` no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER twice for the center and the radius of each circle;
if set to 2, a cursor circle is used for X11 displays and a cursor rectangle for DeAnza systems;
defaulted to 2

`key_flag` K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the rectangle corners for up to 10 circles, the last coord. must be followed by a -1;
defaulted to N

**See also:** DRAW/LINE, DRAW/SLIT, DRAW/ELLIPS, DRAW/RECTANGLE, DRAW/ARROW DRAW/CROSS, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of the circle. To get out, switch both cursors off and press ENTER.
For X11 displays, the cursor circle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw the circle or to exit.
Due to the redrawing of the cursor circle the underlying circle will not have immediately the desired color, but that will be corrected, once you exit from the command.

**Examples:** `DRAW/CIRCLE`
Use cursor circle to define position and size and draw a white circle.

`DRAW/CIRCLE coords F F 100`
Use values from columns :xstart, :ystart, :xend, :yend of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw filled circles with intensity 100 for DeAnza (i.e. gray) or white color for X11.

`DRAW/CIRCLE 200,200,300,300`
Draw a fitting circle into the rectangle with lower left corner at screen pixel (200,200) and upper right corner at (300,300).

# DRAW/CROSS

## DRAW/CROSS *core* 22-JUL-1991 KB

**Purpose:** Draw crosses in the overlay channel.

**Syntax:** `DRAW/CROSS [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

| | |
|---|---|
| `in_spec` | input specification for centerpoint;<br>(a) CURSOR if coordinates are chosen via the cursor-1;<br>(b) name of table containing coordinates in columns labeled :X_COORD and :Y_COORD;<br>(c) x1,y1 defining the coords. of center point;<br>(d) name of integer keyword holding the coords, max. 40 values, i.e. 20 crosses, see parameter 'key_flag'<br>defaulted to CURSOR |
| `coord_ref` | F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;<br>defaulted to S, also any "strange" input is interpreted as "S" ! |
| `draw_opt` | length of arms of cross in screen pixels; defaulted to 3 |
| `intens` | intensity(color),rotation angle (in degrees) of drawing;<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255,0.<br>for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE,0.<br>the line-graph is first built up from the specs and in the very end rotated counter-clockwise by the rotation angle around the center |
| `nocurs` | no. of cursors to use, if you specify 2, cursor rectangle defines the center and the size of the arms - overrides the size given in 'draw_opt';<br>defaulted to 1 |
| `key_flag` | K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the positions for up to 10 lines, the last coord. must be followed by a -1;<br>defaulted to N |

**See also:** DRAW/LINE, DRAW/CIRCLE, DRAW/ELLIPS, DRAW/RECTANGLE, DRAW/ARROW DRAW/SLIT, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with cursor -1 on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of a cross. To get out, switch both cursors off and press ENTER.
For X11 displays the cursor is moved with the mouse. Use the ENTER and EXIT buttons on the mouse to draw or to exit.
Due to the redrawing of the cursor some parts of the cross will not have immediately the desired color, but that will be

**Examples:** `DRAW/CROSS`
Use cursor to define center and draw a white cross with arms of size = 3 screen pixels.

`DRAW/CROSS coords F 10 100`
Use values from columns :x_coord, :y_coord of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw crosses of size 10 with intensity 100 for DeAnza (i.e. gray) or white color for X11.

`DRAW/CROSS 200,200 dropt=5`
Draw a cross with center at screen pixel (200,200) of size 5.

# DRAW/ELLIPSE

## DRAW/ELLIPSE *core* 23-JUL-1991 KB

**Purpose:** Draw ellipses in the overlay channel.

**Syntax:** `DRAW/ELLIPSE [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

`in_spec` input specification for lower left and upper right corner of the ellipses;
(a) CURSOR if coordinates of a rectangle surrounding the required ellipse are chosen via the cursor rectangle;
(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND;
(c) x1,y1,x2,y2 defining the coords. of lower left and upper right corner of a rectangle surrounding the required ellipse;
(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 ellipses, see parameter 'key_flag'
defaulted to CURSOR

`coord_ref` F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;
defaulted to S, also any "strange" input is interpreted as "S" !

`draw_opt` F(ill) or N(ofill) the ellipses;
defaulted to N, also any "strange" input is interpreted as "N" !

`intens` intensity(color),rotation angle (in degrees) of drawing;
for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255.
for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE.
the ellipse is first built up from the specs and in the very end rotated counter-clockwise by the rotation angle around its center

`nocurs` no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER twice for the lower-left and upper-right corner of the surrounding rectangle for each ellipse;
defaulted to 2

`key_flag` K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the rectangle corners for up to 10 ellipses, the last coord. must be followed by a -1;
defaulted to N

**See also:** DRAW/LINE, DRAW/CIRCLE, DRAW/SLIT, DRAW/RECTANGLE, DRAW/ARROW DRAW/CROSS, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of the ellipse. To get out, switch both cursors off and press ENTER.
For X11 displays, the cursor rectangle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw the ellipse or to exit.
Due to the redrawing of the cursor rectangle some parts of the ellipse will not have immediately the desired color, but that will be corrected, once you exit from the command.

**Examples:** `DRAW/ELLIPSE`
Use cursor rectangle to define corners and draw a white ellipses.

`DRAW/ELLIPSE coords F F 100,45.`
Use values from columns :xstart, :ystart, :xend, :yend of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw filled ellipses with intensity 100 for DeAnza (i.e. gray) or white color for X11;
finally rotate the ellipses counter-clockwise by 45.0 degrees around their center.

`DRAW/ELLIPSE 200,200,300,300`
Draw a fitting ellipses into the rectangle with lower left corner at screen pixel (200,200) and upper right corner at (300,300).

# DRAW/IMAGE

## DRAW/IMAGE *core* 22-JAN-1993 KB

**Purpose:** Draw intensities of a line of an image into a display channel.

**Syntax:** `DRAW/IMAGE frame [chanl] [scale] [center] [cuts] [over] [iaux] [fix]`

| | |
|---|---|
| `frame` | name of image frame |
| `chanl` | image display channel no.; defaulted overlay channel |
| `scale` | scaling in x-dimension of line in image frame which is drawn, (meaning like in LOAD/IMAGE); defaulted to 1 |
| `center` | 'xcent,yline' the coordinates of center x-pixel and line (if 2-dim frame) for drawing, for central x-pixel you can also enter the character 'C';<br>defaulted to 'c,@1' (x-center pixel and 1. line) |
| `cuts` | 'lo,hicut' the cut values for intensity scaling, if omitted the cut values from descriptor LHCUTS are used |
| `over` | Y(es), draw over previously loaded image without clearing;<br>if N(o), clear channel first and then draw;<br>default = Y (also any input other than N is interpreted as Y !) |
| `iaux` | 'yscal,scroff,ints,angl' defining the scaling in y, offset of plot from bottom of screen, intensity of plot, and angle of plot with base (in degrees);<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white);<br>for X11 displays it's really a colour (see Note)<br>defaulted to 'y-screensize,0,255,0.0' |
| `fix` | 'framex,screenx' the x-pixel number of the image and the screen which should coincide; i.e. the plot is positioned on the screen in such a way that pixel 'framex' is at screen pixel 'screenx'.<br>the pixels are entered as integers, e.g 'framex' in [1,NPIX(1)] and 'screenx' in [0,x-screensize]<br>If this parameter is given it overrides the center specifications of parameter 'center'! |

**Note:** The parameters described above can also be accessed via FRAME=, CHANL=, SCALE=, CENTER=, CUTS=, OVER=, IAUX=, FIX=
The cut values used for drawing are not written back into descriptor LHCUTS .
The following colours are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8) - "colour" 0 is used to erase. All values in [9,255] are interpreted as White.

**See also:** DRAW/..., LOAD/IMAGE, CLEAR/OVERLAY, SET/OVERLAY

**Examples:** `DRAW/IMAGE cabra`
Draw first line of image 'cabra.bdf' with central x-pixel in screen center into overlay channel, use existing cut values.

`DRAW/IMAGE paloma 0 cent=c,@100 iaux=200,300`
Draw line no. 100 of 2-dim image 'paloma.bdf' in channel 0 with central x-pixel in screen center and the plot starting in line 300 on screen.

DRAW/LINE *core* 22-JUL-1991 KB

**Purpose:** Draw straight lines in the overlay channel.

**Syntax:** `DRAW/LINE [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

| | |
|---|---|
| `in_spec` | input specification for drawing start- and endpoint;<br>(a) CURSOR if coordinates are chosen via the cursor rectangle;<br>(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND;<br>(c) x1,y1,x2,y2 defining the coords. of start and end of line<br>(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 lines, see parameter 'key_flag'<br>defaulted to CURSOR |
| `coord_ref` | F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;<br>defaulted to S (also any other input is interpreted as "S" !) |
| `draw_opt` | direction of line, only used, if cursor rectangle input:<br>RU for left_down to right_up, RD for left_up to right_down in the cursor rectangle;<br>defaulted to RU (also any other input is interpreted as "RU" !) |
| `intens` | intensity(color),rotation angle (in degrees) of drawing;<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255,0.<br>for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE,0.<br>the line-graph is first built up from the specs and in the very end rotated counter-clockwise with the rotation angle |
| `nocurs` | no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER for the start- and endpoint of each line - overrides the direction given in 'draw_opt';<br>defaulted to 2 |
| `key_flag` | K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the positions for up to 10 lines, the last coord. must be followed by a -1; .<br>defaulted to N |

**See also:** DRAW/SLIT, DRAW/CIRCLE, DRAW/ELLIPS, DRAW/RECTANGLE, DRAW/ARROW DRAW/CROSS, DRAW/ANY

# DRAW/LINE

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Magenta(7), Cyan(8).
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of an line. To get out, switch both cursors off and press ENTER.
For X11 displays the cursor rectangle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw or to exit.
Due to the redrawing of the cursor rectangle the endpoints of the line will not have immediately the desired color, but that will be corrected, once you exit from the command.

**Examples:** `DRAW/LINE`
Use cursor rectangle to define endpoints and draw a white line pointing to the upper right corner.

`DRAW/LINE coords F ? 100,45.`
Use values from columns :xstart, :ystart, :xend, :yend of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw lines with intensity 100 for DeAnza (i.e. gray) or white color for X11;
finally rotate the lines around their center by 45.0 degrees.

`DRAW/LINE 200,200,300,300`
Draw an line from screen pixels (200,200) to (300,300).

DRAW/RECTANGLE *core* 23-JUL-1991 KB

**Purpose:** Draw rectangles in the overlay channel.

**Syntax:** `DRAW/RECTANGLE [in_spec] [coord_ref] [draw_opt] [intens] [nocurs] [key_flag]`

`in_spec` input specification for lower left and upper right corner of the rectangle;
(a) CURSOR if coordinates are chosen via the cursor rectangle;
(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND;
(c) x1,y1,x2,y2 defining the coords. of lower left and upper right corner of the rectangle;
(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 rectangles, see parameter 'key_flag'
defaulted to CURSOR

`coord_ref` F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;
defaulted to S, also any "strange" input is interpreted as "S" !

`draw_opt` F(ill) or N(ofill) the rectangle;
defaulted to N, also any "strange" input is interpreted as "N" !

`intens` intensity(color),rotation angle (in degrees) of drawing;
for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255,0.
for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE,0.
the rectangle is first built up from the specs and in the very end rotated counter-clockwise by the rotation angle around its center

`nocurs` no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER twice for the lower-left and upper-right corner of each rectangle;
defaulted to 2

`key_flag` K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the corners for up to 10 rectangles, the last coord. must be followed by a -1;
defaulted to N

**See also:** DRAW/LINE, DRAW/CIRCLE, DRAW/ELLIPS, DRAW/SLIT, DRAW/ARROW
DRAW/CROSS, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of the rectangle. To get out, switch both cursors off and press ENTER.
For X11 displays the cursor rectangle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw or to exit.
Due to the redrawing of the cursor rectangle the underlying rectangle will not have immediately the desired color, but that will be corrected, once you exit from the command.

# DRAW/RECTANGLE

**Examples:** `DRAW/RECTANGLE`
Use cursor rectangle to define corners and draw a white rectangle.

`DRAW/RECTANGLE coords F F 100,45.`
Use values from columns :xstart, :ystart, :xend, :yend of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw filled rectangles with intensity 100 for DeAnza (i.e. gray) or white color for X11;
finally rotate them by 45 degrees counter-clockwise around their center.

`DRAW/RECTANGLE 200,200,300,300`
Draw a rectangle with lower left corner at screen pixel (200,200) and upper right corner at (300,300).

DRAW/SLIT *core* 23-JUL-1991 KB

**Purpose:** Draw slits in the overlay channel.

**Syntax:** `DRAW/SLIT [in_spec] [coord_ref] [draw_opt] [intens] [nocurs]`
`[key_flag]`

| | |
|---|---|
| `in_spec` | input specification for lower left and upper right corner of the rectangle inside the slit;<br>(a) CURSOR if coordinates are chosen via the cursor rectangle;<br>(b) name of table containing coordinates in columns labeled :XSTART, :YSTART, :XEND and :YEND;<br>(c) x1,y1,x2,y2 defining the coords. of lower left and upper right corner;<br>(d) name of integer keyword holding the coords, max. 40 values, i.e. 10 slits, see parameter 'key_flag'<br>defaulted to CURSOR |
| `coord_ref` | F or S, to indicate that coordinates should be interpreted as screen pixels or frame coordinates in the usual MIDAS syntax, only applicable for option (b) and (c); for option (a) and (d) always screen pixels are used;<br>defaulted to S, also any "strange" input is interpreted as "S" ! |
| `draw_opt` | F(ill) or N(ofill) the slit;<br>defaulted to N, also any "strange" input is interpreted as "N" ! |
| `intens` | intensity(color),rotation angle (in degrees) of drawing;<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white) and the parameter is defaulted to 255,0.<br>for X11 displays it's really a color (see Note) and the par. is defaulted to WHITE,0.<br>the slit is first built up from the specs and in the very end rotated counter-clockwise by the rotation angle around its center |
| `nocurs` | no. of cursors to use, if you specify 1, only one cursor is used and you have to press ENTER twice for the corners of each inner slit rectangle;<br>defaulted to 2 |
| `key_flag` | K(ey) or N(okey) to indicate that 'in_spec' holds the name of an integer keyword which contains the inner corners for up to 10 slits, the last coord. must be followed by a -1;<br>defaulted to N |

**See also:** DRAW/LINE, DRAW/CIRCLE, DRAW/ELLIPS, DRAW/RECTANGLE, DRAW/ARROW DRAW/CROSS, DRAW/ANY

**Note:** The parameters may also be cross referenced via
INSPEC=, COOREF=, DROPT=, INTENS=, NOCURS= and KEY=
The following colors are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8). All values in [9,255] are interpreted as White.
If coord_ref = F, a frame must be loaded in the displayed channel.
For DeAnza displays with input_spec = CURSOR, the cursor box has to be set up with both cursors on, TRACK off and RATE on. Pressing the ENTER button on the cursor box will result in the drawing of the slit. To get out, switch both cursors off and press ENTER.
For X11 displays the cursor rectangle is moved with the mouse and adjusted in size with the arrow keys. Use the ENTER and EXIT buttons on the mouse to draw or to exit.
Due to the redrawing of the cursor rectangle some parts of the slit will not have immediately the desired color, but that will be corrected, once you exit from the command.

# ECHO/FULL

**Examples:** `DRAW/SLIT`
Use cursor rectangle to define corners and draw a white IUE slit.

`DRAW/SLIT coords F N 100`
Use values from columns :xstart, :ystart, :xend, :yend of table 'coords.tbl', interpret them as real world coordinates of the currently displayed frame and draw IUE slits with intensity 100, do not fill them.

`DRAW/SLIT 200,200,300,300 S F 128,32.5`
Draw a slit which has been rotated by 32.5 degrees (before the rotation the slit was defined by the screen pixels (200,200) and (300,300) ), fill it with intensity 128.

---

## ECHO/FULL *core* 04-JAN-1988 KB

**Purpose:** Show substitutions in MIDAS procedure files.

**Syntax:** `ECHO/FULL [levla,levlb]`

`levla,levlb` procedure levels interval for full echo;
defaulted to level 1, if entered from terminal;
defaulted to current level, if executed within a MIDAS procedure;
may also be set to ALL to indicate all levels

**See also:** ECHO/OFF, ECHO/ON, DEBUG/PROCEDURE, DEBUG/MODULE
Chapter 3 in MIDAS User Manual, Volume A

**Note:** As ECHO/ON, but after any parameter or variable substitution has taken place, the command line is displayed again.

**Examples:** `ECHO/FULL 2,4`
Turn full echo on for levels 2, 3 and 4 .

---

## ECHO/OFF *core* 04-JAN-1988 KB

**Purpose:** Suppress display of input from MIDAS procedure files.

**Syntax:** `ECHO/OFF [levla,levlb]`

`levla,levlb` procedure levels;
defaulted to level 1, if entered from terminal;
defaulted to current level, if executed within a MIDAS procedure;
may also be set to ALL to indicate all levels

**See also:** ECHO/ON, ECHO/FULL
Chapter 3 in MIDAS User Manual, Volume A

**Note:** The command lines from procedures at levels specified by the command above are not displayed on the terminal and also not stored in the logfile.

**Examples:** `ECHO/OFF all`
Turn echoing off for all procedure levels.

ECHO/ON *core* 04-JAN-1988 KB

**Purpose:** Display input from MIDAS procedure files.

**Syntax:** `ECHO/ON [levla,levlb]`

`levla,levlb` procedure levels;
defaulted to level 1, if entered from terminal;
defaulted to current level, if executed within a MIDAS procedure;
may also be set to ALL to indicate all levels

**See also:** ECHO/OFF, ECHO/FULL
Chapter 3 in MIDAS User Manual, Volume A

**Note:** The command lines from procedures at levels specified by the command above are displayed on the terminal and also stored in the logfile.

**Examples:** `ECHO/ON 2,2`
Enable command echoing at procedure level 2.

# EDIT/TABLE

## EDIT/TABLE *core* 15-MAY-1990 JDP

**Purpose:** To edit table files in MIDAS format. EDT compatible.

**Syntax:** `EDIT/TABLE table [edit_option] [col] [row]`

| | |
|---|---|
| `table` | name of table; if the table doesn't exist and the parameters col and row are provided a new table will be created |
| `edit_option` | r for Read-Only (all modifications are ignored) i for Immediate Update (can't quit....) The table is opened in Input-Output mode per default |
| `col` | number of columns of the table to be created |
| `row` | number of rows of the table to be created |

**Note:** If the table doesn't exist and if the third and fourth parameter are provided, the table will create a new table containing R*4 columns with a default label. The table editor is a modified version of the EDT editor in keypad mode.
All functions of the table editor are also available in command mode which can be enabled by hitting the CNTL Z.
On-line HELP facility available inside the editor by entering the command mode and typing help.
The following functions are available:
CREATE creates a new column
DELETE deletes a column
CHANGE changes the format associated to a column
SORT sorts table according to the values of a column
FIND finds a value in a column
TOP moves cursor to the top of the table
BOTTOM moves cursor to the bottom of a table
RIGTHPAGE displays the following page of a table
LEFTPAGE displays the leftmost page of a table
ROW moves cursor to one specific row of the table
QUIT quit the editor without saving modifications
EXIT exit the editor saving modifications

WARNING: A table shouldn't be opened in read-write mode simultaneously in two parallel Midas sessions. This manipulation could destroy it!!!!!!

**See also:** CREATE/TAB

**Examples:** `EDIT/TABLE daniel`
Edit interactively the existing table daniel.tbl.

`EDIT/TABLE SAO R`
Edit the table SAO.tbl, in read-only mode.

## EQUALIZE/HISTOGRAM *core* 09-NOV-1994 KB

**Purpose:** Perform histogram equalization and load ITT to apply it.

**Syntax:** `EQUALIZE/HISTOGRAM frame [descr] [itt_name] [no_load]`

| | |
|---|---|
| `frame` | name of data frame, defaulted to currently displayed frame |
| `descr` | name of descriptor where the equalized histogram will be stored; default is HIST_EQ |
| `itt_name` | name for optional ITT where to save the resulting ITT; defaulted to no ITT |
| `no_load` | NO if you don't want to load the ITT into the display; defaulted to YES |

**See also:** LOAD/ITT, GET/ITT, MODIFY/ITT, STATISTICS/IMAGE

**Note:** The algorithm uses a histogram of 256 bins stored in the relevant descriptors of the frame, as written e.g. by the command
STATISTICS/IMAGE frame .
If the histogram is "ill formed", e.g. the median is located already in the lowest bin, meaningful equalization cannot be done and the 'ramp' ITT is produced. ALso, a message is displayed and keyword PROGSTAT(5) set to -1, else (i. e. in case of successful equalization) keyword PROGSTAT(5) = 0.
If parm. 'no_load' is not set to NO, the equalized histogram is sent directly as an ITT to the currently displayed channel.

**Examples:** `EQUALIZE/HISTO chango`
Use frame 'chango.bdf' to create an equalized histogram, build an ITT from it and apply it to the image display.

`EQUALIZE/HISTO chango ?  equal no`
Create an equalized histogram as above, build an ITT from it and save that ITT in table 'equal.itt'.

# EXECUTE/CATALOG

## EXECUTE/CATALOG *core* 28-JAN-1994 KB

**Purpose:** Execute a MIDAS procedure or MIDAS command for all entries in a catalog.

**Syntax:** `EXECUTE/CATALOG com_string parm1 ... parm7`

`com_string` name of MIDAS procedure, with type (.prg), or a MIDAS command.

`parm1 ... parm7` up to 7 parameters,
passed to the procedure or MIDAS command

**See also:** WRITE/SETUP, READ/SETUP, INFO/SETUP, INITIALIZE/SETUP
CREATE/ICAT, READ/ICAT, STORE/FRAME

**Note:** At least one of the parameters P2, ..., P8 of EXECUTE/CATALOG has to be of the form 'xyz.cat' to indicate an input catalog.
If output images are created, put a catalog name (with .cat) as the relevant parameter and specify that parameter no. via the "WRITE/SETUP catalog" command. Note, that only a single output catalog is supported by EXECUTE/CATALOG. If that is not suitable, you have to edit your own MIDAS procedure using STORE/FRAME to obtain that functionality.
There are different ways to construct the name of the output frames from the input frame names, also an interval of entry numbers for the first input catalog may be set up. Finally, input frames may be automatically deleted after having been processed.
All that is controlled via the Setup mechanism in MIDAS.
The name of the setup for EXECUTE/CATALOG is 'catalog'.
WRITE/SETUP CATALOG modifies all control variables,
INIT/SETUP CATALOG sets all variables to their default value,
READ/SETUP CATALOG displays the current status and
INFO/SETUP CATALOG explains the syntax of the WRITE/SETUP command for the catalog Setup in detail.
Currently, EXECUTE/CATALOG operates only on image catalogs!

**Examples:** `INFO/SETUP CATALOG`
Get a detailed explanation of all the variables involved.

`WRITE/SETUP catalog 1,4 1 no 1,100 2 zz`
Setup the control variables for all subsequent EXECUTE/CATALOG commands (for explanation use INFO/SETUP CATALOG).

`EXECUTE/CATALOG compute/ima res.cat = aguila.cat + 12.345`
Execute the command COMPUTE/IMAGE for all images with entries in catalog 'aguila.cat'. Add entries for all result images to catalog 'res.cat'.
Note, that parameter no. 6 of the WRITE/SETUP command above was 2, indicating that parameter no. 2 of the EXECUTE/CATALOG command holds the name of the output catalog.

`WRITE/SETUP catalog 1,4 1 no 1,100 0`
Setup the control variables such that no output catalog is expected in a following EXECUTE/CATALOG command.

`EXECUTE/CATALOG READ/DESCR tortuga.cat npix`
Read the descriptor NPIX of all image frames with entries in the catalog 'tortuga.cat'. Note, that the READ/DESCR command does not create an output image.

EXECUTE/TABLE *core* 23-DEC-1993 KB

**Purpose:** Execute a MIDAS command with operands in columns of a MIDAS table looping over all rows of the table.

**Syntax:** `EXECUTE/TABLE table command-string`

`table` name of MIDAS table

`command-string` any MIDAS command using operands which are stored in columns of the table; they are indicated by the column label enclosed in square brackets [ and ], e.g. [:X_COORD]

**See also:** EXECUTE/CATALOG

**Note:** The command creates a Midas procedure 'midtabXY.prg' (with XY = Midas unit) which loops over all rows of the table. This procedure is then executed.

**Examples:** `EXECUTE/TABLE incahuasi compute/image [:image1] = [:image2]+1.234`
Execute the COMPUTE/IMAGE command with result frame names stored in column labeled :IMAGE1 and input image names stored in column labeled :IMAGE2. Execute the command for all rows of the table 'incahuasi.tbl'.

`EXECUTE/TABLE coquimbo write/out [:Ident]`
Write out the contents of column :IDENT of table 'coquimbo.tbl'.

# EXTRACT/CTRACE

## EXTRACT/CTRACE *core* 25-JAN-1995 KB

**Purpose:** Extract interactively a column from a displayed image.

**Syntax:** `EXTRACT/CTRACE [step] [frame] [plot_flag] [zw_option]`

| | |
|---|---|
| `step` | stepsize along the traced column;<br>defaulted to stepsize in y of displayed image |
| `frame` | name for result frame; defaulted to 'column.bdf'. |
| `plot_flag` | DRAW, PLOT or NONE; defaulted to DRAW;<br>if set to DRAW, a plot of the extracted column is displayed in the overlay channel each time the ENTER button is pushed;<br>if set to PLOT, a plot of the extracted column is displayed on the graphics terminal (graphics window) each time the ENTER button is pushed. |
| `zw_option` | zoom_window_flag,zoom;<br>zoom_window_flag = W for zoom_window, N for none,<br>zoom = initial zoom factor; defaulted to N,4;<br>only applicable to X11 window displays! |

**See also:** EXTRACT/TRACE, EXTRACT/RTRACE, EXTRACT/ROTATED
VIEW/IMAGE, GET/CURSOR

**Note:** If you work with a DeAnza then:
Use the joystick/trackball to mark the column in the currently displayed image. To extract the chosen column and exit, set the cursor off and press the ENTER button.
Else if you work with an X Window system then:
Use the mouse to move cursor 0 to mark the column in the currently displayed image. To extract the chosen column and exit, press the EXIT button (the button to the right of the ENTER button).
With the zoom window option you first select a subwindow in the main display window. Then move the mouse into the zoom window and choose the column trace. Push Enter to extract the column and draw the full column vector in the main display window. Press Exit to go back to the main window and choose another subwindow there or press Exit to leave the command.
All the additional functionality which comes with the zoom window option (see e.g. VIEW/IMAGE) is also available.
The parameter 'frame' may be set to "+" also, which means that no frame is extracted (only useful with plot_flag = DRAW or PLOT).

**Examples:** `EXTRACT/CTRACE ? gato`
Extract column of displayed frame and store in 1-dim frame 'gato.bdf' and show the profile, each time the ENTER button is pressed. Use stepsize in y-direction of displayed frame.

`EXTRACT/CTRACE 1.05`
Extract 1-dim frame 'column.bdf' from displayed frame and use stepsize = 1.05 in the sampling of the column.

`EXTRACT/CTRACE ? mosca plot w`
Extract column from displayed frame and store in frame 'mosca.bdf'. Choose the column in the zoom window and plot the profile in the graphics window each time you press Enter (in the zoom window). If no graphics and/or zoom window exist, they are created with default sizes.

## EXTRACT/CURSOR *core* 23-FEB-1994 KB

**Purpose:** Extract a subframe from the frame currently displayed on the ImageDisplay.

**Syntax:** `EXTRACT/CURSOR [subfr] [xpx,ypx] [loop_flag]`

| | |
|---|---|
| `subfr` | name of extracted subframe, defaulted to 'subframe.bdf' |
| `xpx,ypx` | no. of pixels in x, y dimension.<br>If omitted, a cursor rectangle will be used to determine the center and the actual size of the subframe.<br>If given, a single cursor will be used to determine the center of the subframe, its dimensions being determined by xpx,ypx and the actual size of the extracted subframe will be drawn in the overlay plane. |
| `loop_flag` | Loop or NoLoop, if set to Loop more than one subframe can be extracted (until you press the ExitButton), then the param. 'subfr' is used as the root for creating names like subfr0001.bdf, subfr0002.bdf, ... for the extracted frames;<br>defaulted to NoLoop |

**See also:** EXTRACT/IMAGE, EXTRACT/LINE, EXTRACT/SLIT, EXTRACT/ROTATED_IMAGE GET/CURSOR

**Note:** Warning: if the name of the subframe is equal to the name of the displayed frame, this frame will be overwritten.
For a description of how to move and modify the cursor rectangle use HELP [ImageDisplay].

**Examples:** `EXTRACT/CURS delicias`
Extract from displayed frame the subframe indicated by the cursor rectangle and store it into frame 'delicias.bdf'

`EXTRACT/CURS torreon 100,12`
Extract from displayed frame the subframe centered at cursor position, size will be 100 pixels in x-direction and 12 pixels in y-direction; finally store it into 'torreon.bdf'.

`EXTRACT/CURS juarez 8,8 LOOP`
Extract from displayed frame the subframes juarez0001.bdf, juarez0002.bdf, ... each centered at current cursor position, and with size = 8x8 pixels.

`EXTRACT/CURS chihuahua`
We assume, that frame 'chihuahua.bdf' is displayed. Proceed as in the first example, but overwrite the frame named 'chihuahua.bdf' (no new file version will be created for VMS)

# EXTRACT/IMAGE

## EXTRACT/IMAGE *core* 28-JUN-1995 KB

**Purpose:** Extract a subimage from an image frame.

**Syntax:** `EXTRACT/IMAGE subframe = frame[...:...]`

`EXTRACT/IMAGE subframe = frame[...] loffsets roffsets`

| | |
|---|---|
| `subframe` | result frame |
| `frame[...:...]` | frame name + pixel interval<br>or<br>frame[...] = frame name + central pixel |
| `loffsets` | xleft,yleft,zleft - the number of pixels to the "left" of the central pixel |
| `roffsets` | xright,yright,zright - the number of pixels to the "right" of the central pixel |

**Note:** The brackets [..] following the input frame do not indicate an optional parameter, but define the range of extraction:
[c1,c2,..:d1,d2,..]
where ci and di are the ith start and end coordinates in the input frame; ci or di can be in any of the following formats:
a number, to indicate real world coords.
a number preceded by @, to indicate a pixel no.
the symbols. "<" and ">", to indicate start or end pixel.
Please, note, that the bracket follows the frame name directly, no spaces in between!
If the central frame pixel option is used, the start and end pixels of the subframe to be extracted are calculated by subtracting the left offsets from the center and adding the right offsets to it.

**See also:** INSERT/IMAGE, EXTRACT/CURSOR, EXTRACT/LINE

**Note:** The user cuts (i.e. descriptor LHCUTS(1,2)) of subframe, the extracted frame, will remain the same as the input frame.

**Examples:** `EXTRACT/IMAGE out = in[@20,@10:@119,@19]`
Extract subimage of 100 x 20 pixels from the 2-dim frame 'in.bdf' and store it as frame 'out.bdf'.

`EXTRACT/IMAGE out = in[15.1:>]`
Extract subimage from 1-dim frame 'in.bdf'. First pixel in 'out.bdf' is the pixel in 'in.bdf' with world coordinate closest

`EXTRACT/IMAGE out = in[@101,@101] 100,40 22,39`
Extract subimage from 2-dim frame 'in.bdf'. First pixel in 'out.bdf' is the pixel 1,61 and last pixel is 123,140. So 'out.bdf' has 123x80 pixels.

## EXTRACT/LINE *core* 14-FEB-1985 KB

**Purpose:** Extract a 1-dim line from a 2-dim frame.

**Syntax:** `EXTRACT/LINE out = in[..] [step]`

| | |
|---|---|
| `out` | resulting 1-dim. frame |
| `in` | input frame [..] defines "lower left" and "upper right" corners in the frame the resulting frame will be the diagonal from corner to corner |
| `step` | stepsize along the line, default is the stepsize in x of the input frame |

**Note:** The brackets [..] following the input frame do not indicate an optional parameter, but define the range of extraction; [c1,c2,..:d1,d2,..] where ci and di are the ith start and end coordinates in the input frame; ci or di can be in any of the following formats:
a number, to indicate real world coords.
a number preceded by @, to indicate a pixel no.
the chars. "<" and ">", to indicate start or end pixel.
If the name of the result frame is equal to the name of the input frame, this frame will be overwritten.

**See also:** EXTRACT/IMAGE, EXTRACT/TRACE

**Examples:**

```
EXTRACT/LINE sub1 = toro[2.15,1.0:12,20.2] 3.4
```

```
EXTRACT/LINE sub2 = toro[600,@2:800,@5] ?
```

```
EXTRACT/LINE toro = toro[600,@2:800,@5] ?
```

proceed as in the example above, but overwrite the frame named toro.bdf, no new file version will be created (for VMS)

# EXTRACT/REFERENCE_IMAGE

## EXTRACT/REFERENCE_IMAGE *core* 04-JUNE-1986 KB

**Purpose:** Extract subimage according to reference image.

**Syntax:** `EXTRACT/REFERENCE_IMAGE in ref out [thresh]`

| | |
|---|---|
| `in` | input frame |
| `ref` | reference frame, must have same dimension as input frame |
| `out` | resulting 1-dimensional frame |
| `thresh` | threshold in reference frame, defaulted to 1.0 |

**Note:** For all pixels in the ref. frame with a value .GE. 'thresh', the corresponding pixels of the input frame are copied into the result frame, which will be a 1-dim image (i.e. the pixels will be stored as one vector. By changing the descriptors NAXIS, NPIX, etc. you can then adapt the result frame to your needs.

**See also:** EXTRACT/IMAGE, EXTRACT/LINE

**Examples:** `EXTRACT/REFERENCE itest ref result 2.2`
Frame result.bdf will contain all pixels of itest.bdf where the corresponding pixel in ref.bdf has a value .GE. 2.2
`EXTRACT/REFERENCE itest itest result 2.2`
Frame result.bdf will contain all pixels of itest.bdf with a value .GE. 2.2

## EXTRACT/ROTATED_IMAGE *core* 20-SEP-1990 KB

**Purpose:** Extract a rotated subimage from displayed image.

**Syntax:** `EXTRACT/ROTATED_IMAGE [steps] [frame]`

| | |
|---|---|
| `steps` | stepsize in x and y for the extracted image, default is stepsize in x and y of displayed image |
| `frame` | name for result frame, default is 'subframe.bdf' |

**Note:** If you work with a DeAnza then: The cursor box has to be set up initially as: both cursors on, TRACK off and RATE on Use the joystick, to define the base line of the subimage across the currently displayed image. Set both cursors off + press ENTER to fix this line. Then use only cursor 1 to define the corners of the rotated subimage.
Else if you work with a X Window system then: Use the mouse to move the cursor to define the base line of the subimage across the currently displayed image. Press the ENTER button to fix this line. Then use the cursor (mouse) to define the corners of the rotated subimage.
If the baseline is parallel to one axis, the command will abort - you should use EXTRACT/CURSOR or EXTRACT/IMAGE for this case.

**See also:** EXTRACT/CURSOR, EXTRACT/IMAGE

**Examples:** `EXTRACT/ROT 0.4,0.4 zopilote`
Extract a rotated subimage from the displayed image by using the interactively defined endpoints and resampling (using stepsize of 0.4 for x and y) along the lines of the subimage via bilinear interpolation. Result frame will be zopilote.bdf.

EXTRACT/RTRACE *core* 25-JAN-1995 KB

**Purpose:** Extract interactively a row from a displayed image.

**Syntax:** `EXTRACT/RTRACE [step] [frame] [plot_flag] [zw_option]`

| | |
|---|---|
| `step` | stepsize along the traced row;<br>defaultd to stepsize in x of displayed image |
| `frame` | name for result frame; defaulted to 'row.bdf'. |
| `plot_flag` | DRAW, PLOT or NONE; defaulted to DRAW;<br>if set to DRAW, a plot of the extracted row is displayed in the overlay channel each time the ENTER button is pushed;<br>if set to PLOT, a plot of the extracted row is displayed on the graphics terminal (graphics window) each time the ENTER button is pushed; |
| `zw_option` | zoom_window_flag,zoom;<br>zoom_window_flag = W for zoom_window, N for none,<br>zoom = initial zoom factor; defaulted to N,4;<br>only applicable to X11 window displays! |

**See also:** EXTRACT/TRACE, EXTRACT/CTRACE, EXTRACT/ROTATED
VIEW/IMAGE, GET/CURSOR

**Note:** If you work with a DeAnza then:
Use the joystick/trackball to mark the row in the currently displayed image. To extract the chosen row and exit, set the cursor off and press the ENTER button.
Else if you work with an X Window system then:
Use the mouse to move cursor 0 to mark the row in the currently displayed image. To extract the chosen row and exit, press the EXIT button (the button to the right of the ENTER button).
With the zoom window option you first select a subwindow in the main display window. Then move the mouse into the zoom window and choose the row trace. Push Enter to extract the row and draw the full row vector in the main display window. Press Exit to go back to the main window and choose another subwindow there or press Exit to leave the command.
All the additional functionality which comes with the zoom window option (see e.g. VIEW/IMAGE) is also available.
The parameter 'frame' may be set to "+" also, which means that no frame is extracted (only useful with plot_flag = DRAW or PLOT).

**Examples:** `EXTRACT/RTRACE ? gato`
Extract row of displayed frame and store in 1-dim frame 'gato.bdf' and show the profile, each time the ENTER button is pressed. Use stepsize in x-direction of displayed frame.

`EXTRACT/RTRACE 1.05`
Extract 1-dim frame 'row.bdf' from displayed frame and use stepsize = 1.05 in the sampling of the row.

`EXTRACT/RTRACE ? mosca plot w`
Extract row from displayed frame and store in frame 'mosca.bdf'. Choose the row in the zoom window and plot the profile in the graphics window each time you press Enter (in the zoom window). If no graphics and/or zoom window exist, they are created with default sizes.

# EXTRACT/SLIT

## EXTRACT/SLIT *core* 09-APR-1992 KB

**Purpose:** Extract a subimage defined by a fixed slit from image.

**Syntax:** `EXTRACT/SLIT [in_option] [resframe] [slit_specs]`

`in_option` CURSOR, if subimage will be extracted from displayed image via cursor slit (option 1);
inframe if subimage will be extracted from inframe (option 2);
defaulted to CURSOR

`resframe` name for result frame, defaulted to slit.bdf

`slit_specs` definition of slit size:
(a) slitw,slitl,refcoord stepx,stepy
slitw,slitl,refcoord = width, length of slit in real pixels or world coords; refcoord = X or Y, defaulted to X
stepx,stepy = stepsize along the base line and between different lines, default is stepsize in x and y of displayed image
(b) refslit
refslit = reference frame for slit: slitw,slitl,stepx,stepy will be taken from descriptors NPIX and STEP of frame refslit; and if option 2 is used, also the start values for the extracted slit are taken from descr START
(c) refslit slit_angle
slit_angle = angle (in degrees) of baseline of slit (which is the x-axis) with horizontal axis (counterclockwise);
defaulted to 45.0
(a), (b) are possible only for option 1, (c) only for option 2

**See also:** EXTRACT/CURSOR, EXTRACT/TRACE, EXTRACT/ROTA, EXTRACT/IMAGE

**Note:** If you use the CURSOR option:
For DeAnza: Set both cursors on, TRACK off, RATE on. By toggling the on/off switches for each cursor you can move the cursors individually and thus modify the position of the slit but not it's size. To really extract the subimage under the slit, set both cursors off and press ENTER. For X-Windows: Press ENTER button on the mouse to draw the slit which would be extracted. You can move one cursor via the mouse and the other via the arrow keys. Thus you can modify the position of the slit but not it's size. To really extract the subimage under the slit, press the EXIT button on the mouse.

**Examples:** `EXTRA/SLIT ? sub @22,@100`
Use a slit which is 22 real (frame) pixels wide and 100 pixels long. The size of the slit on the screen depends on the scaling and zooming values of the reference coordinate (= X as default) when the image was loaded.

`EXTRA/SLIT ? sub 22.,100.4,Y`
Use a slit which is 'yw' pixels wide and 'yl' pixels long, yw = 22./y-step, yl = 100.4/y-step of displayed image.

`EXTRA/SLIT ? sub refslit`
Use a slit the size of which is determined from the size of the frame 'refslit.bdf'.

`EXTRA/SLIT in sub slit 33.`
Use image 'in.bdf' as input, 'sub.bdf' as output and the image 'slit.bdf' as reference slit with a slit angle of 33. degrees.

EXTRACT/TRACE *core* 25-JAN-1995 KB

**Purpose:** Extract interactively a line from a displayed image.

**Syntax:** `EXTRACT/TRACE [step] [frame] [plot_flag] [cut_option] [zw_option]`

| | |
|---|---|
| `step` | stepsize along the traced line;<br>defaulted to stepsize in x of displayed image |
| `frame` | name for result frame, defaulted to 'trace.bdf'. |
| `plot_flag` | DRAW, PLOT or NONE, defaulted to DRAW;<br>if set to DRAW, a plot of the extracted column is displayed in the overlay channel each time the ENTER button is pushed<br>if set to PLOT, a plot of the extracted column is displayed on the graphics terminal (graphics window) each time the ENTER button is pushed |
| `cut_option` | C(UT) or N(OCUT) , defaulted to N;<br>for C only extract line bound by the two cursors; for N extract complete line through cursors |
| `zw_option` | zoom_window_flag,zoom;<br>zoom_window_flag = W for zoom_window, N for none,<br>zoom = initial zoom factor; defaulted to N,4;<br>only applicable to X11 window displays! |

**See also:** EXTRACT/CTRACE, EXTRACT/RTRACE, EXTRACT/ROTATED
VIEW/IMAGE, GET/CURSOR

**Note:** If you work with a DeAnza then:
The cursor box has to be set up as: both cursors on, TRACK off and RATE on. Use the joystick/trackball, to define a trace across the currently displayed image. Setting cursors on or off allows you to rotate or move the trace in parallel.
To extract the chosen line and exit, set both cursors off and press the ENTER button.
Else if you work with an X Window system then:
Use the mouse to move cursor 0 and the arrow keys of the keyboard to move cursor 1, to define a trace across the currently displayed image. The trace is actually drawn only after pushing the Enter button (left button on the mouse).
To extract the chosen line and exit, press the EXIT button (the button to the right of the ENTER button).
A slice of the displayed frame along this trace line will be put into the new frame (e.g. trace.bdf).
With the zoom window option you first select a subwindow in the main display window. Then move the mouse into the zoom window together with the arrow keys for cursor 1 choose the trace. Push Enter to extract the trace and draw the full trace vector in the main display window. Press Exit to go back to the main window and choose another subwindow there or press Exit to leave the command.
All the additional functionality which comes with the zoom window option (see e.g. VIEW/IMAGE) is also available.
The parameter 'frame' may be set to "+" also, which means that no frame is extracted (only useful with plot_flag = DRAW or PLOT).

**Examples:** `EXTRACT/TRACE ? gato`
Extract 1-dim frame 'gato.bdf' from displayed frame (complete line passing through both cursors) and show the profile, each time the ENTER button is pressed.

```
EXTRACT/TRACE 1.05 ?  ?  C
```

# FFT/FINVERSE

Extract 1-dim frame 'trace.bdf' from displayed frame (only the line segment between the two cursors), use stepsize = 1.05 when extracting pixels along the line.

```
EXTRACT/TRACE ? mosca plot ?  w
```

Extract 1-dim frame 'mosca.bdf' from displayed frame. Choose the trace points in the zoom window and plot the profile in the graphics window each time you press Enter (in the zoom window). If no graphics and/or zoom window exist, they are created with default sizes.

---

## FFT/FINVERSE *core* 26-OCT-1992 KB

**Purpose:** Compute the inverse discrete Fourier transform of real or complex frame.

**Subject:** Transformation, filter.

**Syntax:** `FFT/FINVERSE inr ini outr outi`

| | |
|---|---|
| `inr` | frame containing the real part of input frame; defaulted to 'fftr.bdf' |
| `ini` | frame containing the imaginary part of input frame,<br>if input frame is only real, omit 'ini' or set it to '?', if you enter values for any of the following parameters;<br>defaulted to 'ffti.bdf' |
| `outr` | frame for real part of result transform; defaulted to 'zztr.bdf' |
| `outi` | frame for imaginary part of result transform; defaulted to 'zzti.bdf' |

**See also:** FFT/FREQUENCY, FFT/FPOWER, FFT/INVERSE, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf' and 'exp_ffti.bdf'.
If the input frames are expanded frames from a previous FFT/FREQU with input files which had dimensions not a power of 2, the frames 'orig_r.bdf' and 'orig_i.bdf' with the original size will be extracted from the results as well.
This command corresponds to a previous FFT/FREQUENCY or FFT/FPOWER command, not a FFT/IMAGE command!

**Examples:**

```
FFT/FINV f1 f2
```

Do inverse FFT for image with real part in frame 'f1.bdf' and imaginary part in 'f2.bdf', store results in frame 'zztr.bdf' and 'zzti.bdf' .

```
FFT/FINV
```

Do inverse FFT for image with real part in frame 'fftr.bdf' and imaginary part in 'ffti.bdf', store results in frame 'zztr.bdf' and 'zzti.bdf' .

**Restrictions:**

Does not work for 3-dim frames.

FFT/FPOWER *core* 26-OCT-1992 KB

**Purpose:** Compute the discrete Fourier transform of real or complex frame and also its power spectrum (square root). Also shift origin of frequency domain to center of image.

**Subject:** Transformation, filter.

**Syntax:** `FFT/FPOWER inr ini outr outi pow_spec`

| | |
|---|---|
| `inr` | frame containing the real part of input frame |
| `ini` | frame containing the imaginary part of input frame; if input frame is real, omit parameter 'ini' or set it to '?' if you enter values for any of the following parameters. |
| `outr` | frame for real part of result transform, defaulted to 'fftr.bdf' |
| `outi` | frame for imaginary part of result transform, defaulted to 'ffti.bdf' |
| `pow_spec` | frame for power spectrum, defaulted to 'power.bdf' |

**See also:** FFT/FINVERSE, FFT/FREQUENCY, FFT/POWER, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf', 'exp_ffti.bdf' and 'exp_pow.bdf'. These frames should be used later when calculating the inverse Fourier transform via FFT/FINVERSE!

**Examples:** `FFT/FPOWER rr ri`
Do FFT for image where real part is contained in frame 'rr.bdf' and imaginary part in frame 'ri.bdf'. The real and imaginary parts of the FFT are stored in 'fftr.bdf' and 'ffti.bdf'; the power spectrum will be in frame 'power.bdf'.

`FFT/FPOWER lola ? zr zi zp`
Do FFT for real frame 'lola.bdf' (no imaginary part) and store results in 'zr.bdf', 'zi.bdf' and 'zp.bdf'.

# FFT/FREQUENCY

## FFT/FREQUENCY *core* 26-OCT-1992 KB

**Purpose:** Compute the discrete Fourier transform of real or complex frame and shift origin of frequency domain to center.

**Subject:** Transformation, filter.

**Syntax:** `FFT/FREQUENCY inr ini outr outi`

| | |
|---|---|
| `inr` | frame containing the real part of input frame |
| `ini` | frame containing the imaginary part of input frame;<br>if input frame is real, omit parameter 'ini' or set it to '?' if you enter values for any of the following parameters. |
| `outr` | frame for real part of result transform, defaulted to 'fftr.bdf' |
| `outi` | frame for imaginary part of result transform, defaulted to 'ffti.bdf' |

**See also:** FFT/FINVERSE, FFT/FPOWER, FFT/IMAGE, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf' and 'exp_ffti.bdf'. These frames should be used later when calculating the inverse Fourier transform via FFT/FINVERSE!

**Examples:** `FFT/FREQU rr ri`
Do FFT for image where real part is contained in frame 'rr.bdf' and imaginary part in frame 'ri.bdf'. The real and imaginary parts of the FFT are stored in 'fftr.bdf' and 'ffti.bdf'.

`FFT/FREQU lola ?  zr zi`
Do FFT for real frame 'lola.bdf' (no imaginary part) and store results in 'zr.bdf' and 'zi.bdf'.

**Restrictions:**
Does not work for 3-dim frames.

FFT/IMAGE *core* 26-OCT-1992 KB

**Purpose:** Compute the discrete Fourier transform of real or complex frame.

**Subject:** Transformation, filter.

**Syntax:** `FFT/IMAGE inr ini outr outi`

| | |
|---|---|
| `inr` | frame containing the real part of input frame |
| `ini` | frame containing the imaginary part of input frame;<br>if input frame is real, omit parameter 'ini' or set it to '?' if you enter values for any of the following parameters. |
| `outr` | frame for real part of result transform, defaulted to 'fftr.bdf' |
| `outi` | frame for imaginary part of result transform, defaulted to 'ffti.bdf' |

**See also:** FFT/INVERSE, FFT/POWER, FFT/FREQUENCY, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf' and 'exp_ffti.bdf'. These frames should be used later when calculating the inverse Fourier transform via FFT/INVERSE!

**Examples:** `FFT/IMA rr ri`
Do FFT for image where real part is contained in frame 'rr.bdf' and imaginary part in frame 'ri.bdf'. The real and imaginary parts of the FFT are stored in 'fftr.bdf' and 'ffti.bdf'.

`FFT/IMA lola ?  zr zi`
Do FFT for real frame 'lola.bdf' (no imaginary part) and store results in 'zr.bdf' and 'zi.bdf'.

**Restrictions:**
Does not work for 3-dim frames.

# FFT/INVERSE

## FFT/INVERSE *core* 26-OCT-1992 KB

**Purpose:** Compute the inverse discrete Fourier transform of real or complex frame.

**Subject:** Transformation, filter.

**Syntax:** `FFT/INVERSE inr ini outr outi`

| | |
|---|---|
| `inr` | frame containing the real part of input frame; defaulted to 'fftr.bdf' |
| `ini` | frame containing the imaginary part of input frame,<br>if input frame is only real, omit 'ini' or set it to '?', if you enter values for any of the following parameters;<br>defaulted to 'ffti.bdf' |
| `outr` | frame for real part of result transform; defaulted to 'zztr.bdf' |
| `outi` | frame for imaginary part of result transform; defaulted to 'zzti.bdf' |

**See also:** FFT/IMAGE, FFT/POWER, FFT/FINVERSE, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf' and 'exp_ffti.bdf'.
If the input frames are expanded frames from a previous FFT/IMAGE with input files which had dimensions not a power of 2, the frames 'orig_r.bdf' and 'orig_i.bdf' with the original size will be extracted from the results as well.
This command corresponds to a previous FFT/IMAGE or FFT/POWER command, not a FFT/FREQUENCY command!

**Examples:** `FFT/INV f1 f2`
Do inverse FFT for image with real part in frame 'f1.bdf' and imaginary part in 'f2.bdf', store results in frame 'zztr.bdf' and 'zzti.bdf' .

`FFT/INV`
Do inverse FFT for image with real part in frame 'fftr.bdf' and imaginary part in 'ffti.bdf', store results in frame 'zztr.bdf' and 'zzti.bdf' .

**Restrictions:**
Does not work for 3-dim frames.

FFT/POWER *core* 26-OCT-1992 KB

**Purpose:** Compute the discrete Fourier transform of real or complex frame and also its power spectrum (square root).

**Subject:** Transformation, filter.

**Syntax:** `FFT/POWER inr ini outr outi pow_spec`

| | |
|---|---|
| `inr` | frame containing the real part of input frame |
| `ini` | frame containing the imaginary part of input frame;<br>if input frame is real, omit parameter 'ini' or set it to '?' if you enter values for any of the following parameters. |
| `outr` | frame for real part of result transform, defaulted to 'fftr.bdf' |
| `outi` | frame for imaginary part of result transform, defaulted to 'ffti.bdf' |
| `pow_spec` | frame for power spectrum, defaulted to 'power.bdf' |

**See also:** FFT/INVERSE, FFT/IMAGE, FFT/FPOWER, CREATE/FILTER

**Note:** All internal calculations are done in double precision and only the final result frames are truncated to single precision.
If the number of pixels in an axis of the input frame is not a power of 2, the input frame is expanded to the next higher power of 2. The result frames will have the original no. of pixels but the result frames with expanded size are also kept with names 'exp_fftr.bdf', 'exp_ffti.bdf' and 'exp_pow.bdf'. These frames should be used later when calculating the inverse Fourier transform via FFT/INVERSE!

**Examples:** `FFT/POWER rr ri`
Do FFT for image where real part is contained in frame 'rr.bdf' and imaginary part in frame 'ri.bdf'. The real and imaginary parts of the FFT are stored in 'fftr.bdf' and 'ffti.bdf'; the power spectrum will be in frame 'power.bdf'.

`FFT/POWER lola ?  zr zi zp`
Do FFT for real frame 'lola.bdf' (no imaginary part) and store results in 'zr.bdf', 'zi.bdf' and 'zp.bdf'.

**Restrictions:**
Does not work for 3-dim frames.

# FILTER/COSMIC

## FILTER/COSMIC *core* 21-NOV-1991 PM,MR

**Purpose:** Remove cosmic ray events on a single CCD image and replace them by a local median value.

**Syntax:** `FILTER/COSMIC inframe outframe sky,gain,ron,[ns,rc] [mask]`

`inframe` raw input frame

`outframe` output frame

`sky,gain,ron,ns,rc`
sky level (sky), inverse gain factor (e-/ADU) (gain), read-out-noise (in ADU) (ron), threshold for the detection of cosmic rays (ns), and critical ratio for discrimination of objects and cosmic rays (rc). The detection threshold is in units of the theoretical noise sigma of each pixel; it's default value is 4. The default for 'rc' is 2.

`mask` name of an optional frame containing the value 1 for cosmic rays and 0 for all other pixels.

**Note:** a) The algorithm works as follows:
1. The input image is filtered in the following way:
FILTER/MEDIAN inframe &a 1,1 NQ For Long-Slit spectra of extended sources, the algorithm may be more efficient if the median filter works only along the slit.
2. The input image is compared with the filtered image. All pixels with an intensity I greater than Im+ns*sigma are suspicious and may be cosmic rays (Im is the filtered intensity of a pixel and sigma is given by: sigma**2 = ron**2+I/gain).
3. All suspicious pixels are grouped into sets of contiguous points. In each of these sets, the pixel with the maximum intensity Imax is selected. If (Imax-sky) is greater than rc*(Iaver-sky), Iaver being an average of the intensities of the first eight neighbours of that pixel, the whole set of points is considered as a cosmic ray event.
4. The intensities of the pixels affected by cosmic rays are replaced by a median value calculated over the nearest neighbours of the group to which they belong.
b) In many situations, rc is the most critical parameter and requires careful fine-tuning. If it is choosen too small, small sources such as stars may be affected. If rc is too large, the filter may not remove weak partical hits superimposed to reasonably well exposed extended sources.

**See also:** FILTER/MEDIAN

**Examples:** `FILTER/COSMIC ccd ccdclean 150,1,4,4,2.3 cosmask`
Apply cosmic filtering to image 'ccd.bdf' and store result frame as 'ccdclean.bdf'; save a 0/1 mask in 'cosmask.bdf'.

FILTER/DIGITAL *core* 07-MAR-1984 KB

**Purpose:** Use digital filter on an image.

**Subject:** Smoothing.

**Syntax:** `FILTER/DIGITAL frame outframe [filter_specs] [subimage] [options]`

`frame` input image

`outframe` result image

`filter_specs` (a) either a list of weights (max. 20), e.g. w(1),w(2),...,w(9) for a 3x3 template numbered

```
1 2 3
4 5 6
7 8 9 .
```

for 1-dim filters in x enter w(1),...,w(n):x for 1-dim filters in y enter w(1),...,w(n):y ($n < 21$);
(b) or name of a predefined filter, LAPLACE, LOW_PASS, POINT CONE, SHARP and TENT are the names of the currently available filters:
LAPLACE (this filter boosts high spatial frequencies)

```
0 -1 0
-1 5 -1
0 -1 0
```

LOW_PASS (this filter boosts low spatial frequencies)

```
1 2 1
2 5 2
1 2 1
```

POINT (this filter boosts point source)

```
-1 -2 -1
-2 13 -2
-1 -2 -1
```

CONE (this filter enhances Gaussian PSF structures)

```
1 -6 1
-6 36 -6
1 -6 1
```

SHARP

```
-1 -2 -1
-2 13 -2
-1 -2 -1
```

TENT

```
1 1 1
1 8 1
1 1 1
```

these filters are internally normalized, i.e. total sum = 1.0.
(c) or name of an image serving as filter, in that case the size of the weight matrix is determined by the size of the filter image; the dimensions in x and y must be odd numbers!
Defaulted to predefined filter POINT.

# FILTER/DIGITAL

`subimage` optional specification of a subimage on which the filter will be applied, may be either
[xstart,ystart:xend,yend] (see page 3.4 in the MIDAS user manual, chapter 3) to indicate the limits of the subimage (with coordinates given according the MIDAS standard), or
CURSOR if the subimage will be chosen interactively.
table_name if the table "table_name" holds the endpoints of the subimage(s) in columns labeled :XSTART, :YSTART, :XEND and :YEND,
Defaulted to the complete input frame.

`options` flag of 3 characters for options;
(1) L or N for loop or no loop,
(2) D or N for display of smoothed area or no display;
(3) E or N for expanding input frame first, so that result frame has same dimensions as input frame or no expansion with result frame smaller than input;
defaulted to NNE

**See also:** FILTER/MAX, FILTER/MIN, FILTER/GAUSS, FILTER/MEDIAN, FILTER/SMOOTH, TUTORIAL/FILTER, FFT/IMAGE, CONVOLVE/IMAGE, DECONVOLVE/IMAGE

**Note:** New points are calculated as outpix = SUM(inpix*w(m)), where the SUM is taken over the pixels in the neighbourhood determined by the size of the filter kernel, i.e. NPIX(1) and NPIX(2) of the digital filter.
If the subimage option is chosen, it must be at least as big as the filter kernel...
If 'option(3:3) = 'E' the input image is expanded by 2*radx columns and 2*rady lines, so that also the "edge" pixels can be calculated.
Note, that the flux is not conserved.

**Examples:** `FILTER/DIGITAL gordo flaco LAPLACE`
Use the Laplacian filter with the weights shown above on image 'gordo.bdf' and store resulting image into frame 'flaco.bdf'.
The input frame is first expanded, so that the dimensions of the frame 'flaco.bdf' are the same as the ones of 'gordo.bdf'.

`FILTER/DIGITAL in out 0.,-2.,0.,-2.,8.,-2.,0.,-2.,0.`
Use the filter with weights

```
0 -2 0
-2 8 -2
0 -2 0
```

on image 'in.bdf' and store resulting image into frame 'out.bdf'. Remember, that the number of weights determines the size of the filter kernel.

FILTER/GAUSS *core* 18-JUL-1995 KB

**Purpose:** Use Gaussian filter on an image.

**Subject:** Image smoothing

**Syntax:** `FILTER/GAUSS in out [radx,rady] [gauss_specs] [subima]`
`[filtnam] [options]`

| | |
|---|---|
| `in` | input image |
| `out` | result image |
| `radx,rady` | radius in x and y. Radius is the number of pixels "around" the central pixel to be included in the neighbourhood. Total no. of pixels in neighbourhood is (2*radx+1)*(2*rady+1); defaulted to 9,9 (= 19*19 neighbourhood) |
| `gauss_specs` | x-mean,x-sigma and y-mean,y-sigma;<br>defaulted to 9,3,9,3 pixels;<br>for 1-dim case specify xmean,xsigma only |
| `subima` | optional specification of a subimage on which the filter will be applied, may be either<br>[xstart,ystart:xend,yend] (see page 3.4 in the MIDAS User manual, chapter 3) to indicate the limits of the subimage (with coordinates given according to the MIDAS standard), or<br>CURSOR if subimage will be chosen interactively;<br>table_name if the table "table_name" holds the endpoints of the subimage(s),<br>defaulted to the complete input frame |
| `filtnam` | name for the Gaussian filter used; if not given the filter is not kept. This option is a debugging tool. |
| `options` | flag of 2 characters for options;<br>(1) L or N for loop or no loop,<br>(2) D or N for display of smoothed area or no display;<br>defaulted to NN |

**See also:** FILTER/DIGITAL, FILTER/SMOOTH, FILTER/MEDIAN, TUTORIAL/FILTER

**Note:** The Gaussian filter will have (2*radx+1)*(2*rady+1) pixels, lower left corner will have start values 0.,0. and stepsize is 1.,1.
New points are calculated as outpix = SUM(inpix*gausspix) where the SUM is taken over the pixels in the neighbourhood determined by radx and rady.
If the subimage option is chosen, it must be at least as big as the Gaussian filter which has (2*radx+1) * (2*rady+1) pixels.
Note, that the flux is not conserved.

**Examples:** `FILTER/GAUSS gordo flaco`
Use the default Gaussian 19*19 filter on image 'gordo.bdf' and store resulting image into frame 'flaco.bdf'.

`FILTER/GAUSS in out 7,7 7,2,7,2 CURSOR`
Use a 15*15 Gaussian filter with x-mean = 7,x-sigma = 2, y-mean = 7, y-sigma = 2 on cursor selected subframe of image 'in.bdf' and store resulting image into frame 'out.bdf'.

# FILTER/MAX

## FILTER/MAX *core* 09-DEC-1992 KB

**Purpose:** Apply a maximum filter to a frame.

**Subject:** Image smoothing, nonlinear filtering.

**Syntax:** `FILTER/MAX frame outfram [xyradius] [subima] [options]`

| | |
|---|---|
| `frame` | input frame |
| `outfram` | result frame |
| `xyradius` | radx,rady where 'radx' and 'rady' are the radius in x and y, i.e. the number of pixels "around" the central pixel which define the neighbourhood (NBH) of that pixel,<br>total no. of pixels in NBH is then: (2*radx+1)*(2*rady+1);<br>defaulted to 1,1 (a 3x3 neighbourhood, i.e. 9 pixel) |
| `subima` | optional specification of a subimage on which the filter will be applied, may be either [xstart,ystart:xend,yend] (see page 3.4 in the MIDAS User manual, chapter 3) to indicate the limits of the subimage (with coordinates given according to the MIDAS standard), or<br>CURSOR if subimage will be chosen interactively, or<br>table_name if the table "table_name" holds the endpoints of the subimage(s) in columns labeled :XSTART, :YSTART, :XEND and :YEND,<br>defaulted to the complete input frame. |
| `options` | flag of 3 characters for options;<br>(1) L or N for loop or no loop,<br>(2) D or N for display of smoothed area or no display;<br>(3) E or N for expanding input frame first, so that result frame has same dimensions as input frame or no expansion with result frame smaller than input<br>defaulted to NNE |

**See also:** FILTER/MIN, FILTER/GAUSS, FILTER/DIGITAL, FILTER/MEDIAN, FILTER/SMOOTH, TUTORIAL/FILTER, FFT/IMAGE, CONVOLVE/IMAGE, DECONVOLVE/IMAGE

**Note:** Smooth the image by replacing each pixel value I(x,y) by Max(x,y), the maximum of all points in the neighbourhood of (x,y), which contains all pixels (x',y') with abs(x-x') .LE. radx and abs(y-y') .LE. rady (radx, rady as given in the 3rd parameter).
The values radx and rady do not have to be equal, by setting radx or rady to 0, also 1-dimensional filters may be used.
If the subimage option is chosen, it must be at least as big as the filter kernel which has (2*radx+1) * (2*rady+1) pixels.
If 'option(3:3) = 'E' the input image is expanded by 2*radx columns and 2*rady lines, so that also the "edge" pixels can be calculated.
Note, that the flux is not conserved.

**Examples:** `FILTER/MAX gordo flaco`
Smooth image 'gordo.bdf' by getting the maximum over a 3*3 window around each pixel and set the corresponding pixels in frame 'flaco.bdf' to that maximum.
The input frame is first expanded, so that the dimensions of the frame 'flaco.bdf' are the same as the ones of 'gordo.bdf'.

`FILTER/MAX gordo flaco 1,1 CURSOR`

As above, but do the filtering process only to subimages of frame 'gordo.bdf' which are chosen via the cursor.
It is assumed, that image 'gordo.bdf' is loaded into the image display!

`FILTER/MAX gordo flaco 3,0 coords`
As above, but the subimage specifications are taken from the table 'coords.tbl'. This table could e.g. be created by using the command GET/CURSOR with the table option (and 2 cursors!) Furthermore, a 7x1 window around each pixel will be used to find the maximum.

`FILTER/MAX gordo flaco 2,1 [@20,<:@50,>]`
As above, but on only the subimage specified in the command line.

`FILTER/MAX gordo flaco ? ? nnn`
As the first example, but the input frame is not expanded. The frame 'flaco.bdf' will have xdim = NPX-2 and ydim = NPY-2, with NPX, NPY the no. of pixels in x,y of the input frame 'gordo.bdf'.

# FILTER/MEDIAN

## FILTER/MEDIAN *core* 17-MAR-1994 KB

**Purpose:** Apply median filter to a frame.

**Subject:** Image smoothing, image defects, nonlinear filtering.

**Syntax:** `FILTER/MEDIAN frame outfram [filt_specs] [flag] [subima] [options]`

`frame` input frame

`outfram` result frame

`filt_specs` 'radx,rady,threshold' or 'radx,rady,i_low,i_hi'
or 'radx,rady,fa,fb';
radx, rady is the radius in x and y, i.e. the number of pixels "around" the central pixel which define the neighbourhood (NBH) of that pixel,
total no. of pixels in NBH is then: (2*radx+1)*(2*rady+1);
threshold is in [0.,1.] (if relative) or >= 0. (if absolute);
i_low, i_hi are the low intensity, high intensity for the fixed interval option (flag(2) = 'F'),
< and > may be used to indicate lowest and highest value.
fa, fb are the factors used in computing the threshold used with flag(2)='Z'
Default: 1,1,0.0

`flag` two character flag,
(1) Y or N, to indicate inclusion of central pixel in the calculation of median over the neighbourhood,
(2) A for absolute threshold,
R for relative threshold using pixel value,
Q for relative threshold using median value,
F for fixed intensity interval,
Z for computing the threshold according to the formula fa + (fb * SQRT(median_value))

Default: YR

`subima` optional specification of a subimage on which the filter will be applied, may be either [xstart,ystart:xend,yend] (see page 3.4 in the MIDAS User manual, chapter 3) to indicate the limits of the subimage (with coordinates given according the MIDAS standard), or
CURSOR if subimage will be chosen interactively, or
table_name if the table "table_name" holds the endpoints of the subimage(s) in columns labeled :XSTART, :YSTART, :XEND and :YEND,
defaulted to the complete input frame

`options` flag of 3 characters for options;
(1) L or N for loop or no loop,
(2) D or N for display of smoothed area or no display;
(3) E or N for expanding input frame first, so that result frame has same dimensions as input frame or no expansion with result frame smaller than input
defaulted to NNE

**See also:** FILTER/MAX, FILTER/MIN, FILTER/GAUSS, FILTER/DIGITAL, FILTER/SMOOTH, TUTORIAL/FILTER, FFT/IMAGE, CONVOLVE/IMAGE, DECONVOLVE/IMAGE

**Note:** Smooth the image by
1) calculating M(x,y), the median of all points in the neighbourhood of (x,y), which contains all pixels (x',y') with abs(x-x') .LE. radx and abs(y-y') .LE. rady (radx, rady as given in the 3rd parameter) So radx = 1 and rady = 1 yield the usual 3*3 neighbourhood.
2) replacing the original value I(x,y) by M(x,y), if we have for ABS = abs(M(x,y)-I(x,y)):
ABS > thresh (flag(2) = A), ABS > thresh*I(x,y) (flag(2) = R), ABS > thresh*M(x,y) (flag(2) = Q),
or if I(x,y) in [intens_low,intens_hi] (flag(2) = F), (thresh and intens_low, intens_hi as given in the 3rd parameter)
The values radx and rady do not have to be equal, by setting radx or rady to 0, also 1-dimensional filters may be used.
If the subimage option is chosen, it must be at least as big as the filter kernel which has (2*radx+1) * (2*rady+1) pixels.
If 'option(3:3) = 'E' the input image is expanded by 2*radx columns and 2*rady lines, so that also the "edge" pixels can be calculated.
Note, that the flux is not conserved.

**Examples:** `FILTER/MEDIAN gordo flaco 1,1,0.5`
For image 'gordo.bdf' calculate median over a 3*3 neighbourhood and replace the original value by the median value only, if median value differs more than 0.5*original value from the original value.

`FILTER/MED gordo flaco 1,1,0.5 YA CURSOR`
Smooth image 'gordo.bdf', get median over a 3*3 neighbourhood and replace the original value by the median value only, if the median value differs by more than 0.5 from the original value, also apply the median filter only to subimages of 'gordo.bdf' which will be chosen via the cursor.
It is assumed, that image 'gordo.bdf' is loaded into the image display!

`FILTER/MED gordo flaco 1,1,0.5 YA coords`
As above, but the subimage specifications are taken from the table 'coords.tbl'. This table could e.g. be created by using the command GET/CURSOR with the table option (and 2 cursors!)

`FILTER/MED gordo flaco 3,0,0. NA [@20,<:@50,>]`
Smooth image 'gordo.bdf', calculate 1-dim median over 7 x-pixels; do not include the central value in the median calculation and always replace the original value by the median value.
Apply the median filter only to the subimage of frame 'gordo.bdf' which consists of the rows between row no. 20 and row no. 50.

`FILTER/MED gordo flaco 3,2,22.3,24.6 NF ? NNN`
Apply median filter over a 7*5 neighbourhood only to pixels which have an intensity in the interval [22.3,24.6]. Do not include the central pixel in the calculation of the median. 'flaco.bdf' will have xdim = NPX-6 and ydim = NPY-4, with NPX, NPY the no. of pixels in x,y of the input frame 'gordo.bdf'.

`FILTER/MED grande chico 3,3,0.9,1.2 YZ`
Apply median filter over a 7*7 neighbourhood and replace the original value by the median value only, if the median value differs by more than 'd' (with d = 0.9+1.2*SQRT(median)) from the original value.

# FILTER/MIN

## FILTER/MIN *core* 09-DEC-1992 KB

**Purpose:** Apply a minimum filter to a frame.

**Subject:** Image smoothing, nonlinear filtering.

**Syntax:** `FILTER/MIN frame outfram [xyradius] [subima] [options]`

| | |
|---|---|
| `frame` | input frame |
| `outfram` | result frame |
| `xyradius` | radx,rady where 'radx' and 'rady' are the radius in x and y, i.e. the number of pixels "around" the central pixel which define the neighbourhood (NBH) of that pixel,<br>total no. of pixels in NBH is then: (2*radx+1)*(2*rady+1);<br>defaulted to 1,1 (a 3x3 neighbourhood, i.e. 9 pixel) |
| `subima` | optional specification of a subimage on which the filter will be applied, may be either [xstart,ystart:xend,yend] (see page 3.4 in the MIDAS User manual, chapter 3) to indicate the limits of the subimage (with coordinates given according to the MIDAS standard), or<br>CURSOR if subimage will be chosen interactively, or<br>table_name if the table "table_name" holds the endpoints of the subimage(s) in columns labeled :XSTART, :YSTART, :XEND and :YEND,<br>defaulted to the complete input frame. |
| `options` | flag of 3 characters for options;<br>(1) L or N for loop or no loop,<br>(2) D or N for display of smoothed area or no display;<br>(3) E or N for expanding input frame first, so that result frame has same dimensions as input frame or no expansion with result frame smaller than input<br>defaulted to NNE |

**See also:** FILTER/MAX, FILTER/GAUSS, FILTER/DIGITAL, FILTER/MEDIAN, FILTER/SMOOTH, TUTORIAL/FILTER, FFT/IMAGE, CONVOLVE/IMAGE, DECONVOLVE/IMAGE

**Note:** Smooth the image by replacing each pixel value I(x,y) by Min(x,y), the minimum of all points in the neighbourhood of (x,y), which contains all pixels (x',y') with abs(x-x') .LE. radx and abs(y-y') .LE. rady (radx, rady as given in the 3rd parameter).
The values radx and rady do not have to be equal, by setting radx or rady to 0, also 1-dimensional filters may be used.
If the subimage option is chosen, it must be at least as big as the filter kernel which has (2*radx+1) * (2*rady+1) pixels.
If 'option(3:3) = 'E' the input image is expanded by 2*radx columns and 2*rady lines, so that also the "edge" pixels can be calculated.
Note, that the flux is not conserved.

**Examples:** `FILTER/MIN gordo flaco`
Smooth image 'gordo.bdf' by getting the minimum over a 3*3 window around each pixel and set the corresponding pixels in frame 'flaco.bdf' to that minimum.
The input frame is first expanded, so that the dimensions of the frame 'flaco.bdf' are the same as the ones of 'gordo.bdf'.

`FILTER/MIN gordo flaco 1,1 CURSOR`

As above, but do the filtering process only to subimages of frame 'gordo.bdf' which are chosen via the cursor.
It is assumed, that image 'gordo.bdf' is loaded into the image display!

`FILTER/MIN gordo flaco 3,0 coords`
As above, but the subimage specifications are taken from the table 'coords.tbl'. This table could e.g. be created by using the command GET/CURSOR with the table option (and 2 cursors!)
Furthermore, a 7x1 window around each pixel will be used to find the minimum.

`FILTER/MIN gordo flaco 2,1 [@20,<:@50,>]`
As above, but on only the subimage specified in the command line.

`FILTER/MIN gordo flaco ? ? nnn`
As the first example, but the input frame is not expanded. The frame 'flaco.bdf' will have xdim = NPX-2 and ydim = NPY-2, with NPX, NPY the no. of pixels in x,y of the input frame 'gordo.bdf'.

# FILTER/SMOOTH

## FILTER/SMOOTH *core* 10-FEB-1992 KB

**Purpose:** Apply a filter using the running average.

**Subject:** Box average, running mean.

**Syntax:** `FILTER/SMOOTH frame outfram [filter_specs] [flag] [subima] [options]`

`frame` input frame

`outfram` result frame

`filter_specs` radx,rady,threshold
or
radx,rady,intens_low,intens_hi;
radx, rady is the radius in x and y, i.e. the number of pixels "around" the central pixel which define the neighbourhood (NBH) of that pixel,
total no. of pixels in NBH is then: (2*radx+1)*(2*rady+1);
threshold is in [0.,1.] (if relative) or >= 0. (if absolute);
intens_low, hi is the low intensity, high intensity for the fixed interval option (flag(2) = 'F'),
< and > may be used to indicate lowest and highest value.
Default: 1,1,0.0

`flag` two character flag,
(1) Y or N, to indicate inclusion of central pixel in the smoothing over the neighbourhood,
(2) A for absolute threshold,
R for relative threshold using pixel value,
Q for relative threshold using smoothed value,
F for fixed intensity interval, Default: YR

`subima` optional specification of a subimage on which the filter will be applied, may be either [xstart,ystart:xend,yend] (see page 3.4 in the MIDAS User manual, chapter 3) to indicate the limits of the subimage (with coordinates given according to the MIDAS standard), or
CURSOR if subimage will be chosen interactively, or
table_name if the table "table_name" holds the endpoints of the subimage(s) in columns labeled :XSTART, :YSTART, :XEND and :YEND,
defaulted to the complete input frame.

`options` flag of 3 characters for options;
(1) L or N for loop or no loop,
(2) D or N for display of smoothed area or no display;
(3) E or N for expanding input frame first, so that result frame has same dimensions as input frame or no expansion with result frame smaller than input
defaulted to NNE

**See also:** FILTER/MAX, FILTER/MIN, FILTER/GAUSS, FILTER/DIGITAL, FILTER/MEDIAN, TUTORIAL/FILTER, FFT/IMAGE, CONVOLVE/IMAGE, DECONVOLVE/IMAGE

**Note:** Smooth the image by
1) calculating A(x,y), the average of all points in the neighbourhood of (x,y), which contains all points (x',y') with abs(x-x') .LE. radx and abs(y-y') .LE. rady (radx, rady as given in the 3rd parameter) So radx = 1 and rady = 1 yields the usual 3*3 neighbourhood.
2) replacing the original value I(x,y) by A(x,y), if we have for ABS = abs(A(x,y)-I(x,y)):
ABS > thresh (flag(2) = A), ABS > thresh*I(x,y) (flag(2) = R), ABS > thresh*A(x,y) (flag(2) = Q),
or if I(x,y) in [intens_low,intens_hi] (flag(2) = F), (thresh and intens_low, intens_hi as given in the 3rd parameter)
The values radx and rady do not have to be equal, by setting radx or rady to 0, also 1-dimensional filters may be used.
If the subimage option is chosen, it must be at least as big as the filter kernel which has (2*radx+1) * (2*rady+1) pixels.
If 'option(3:3) = 'E' the input image is expanded by 2*radx columns and 2*rady lines, so that also the "edge" pixels can be calculated.
Note, that the flux is not conserved.

**Examples:** `FILTER/SMOOTH gordo flaco 1,1,0.5`
Smooth image 'gordo.bdf', calculate average over a 3*3 window around each pixel and replace the original value by the average value only if average value differs more than 0.5*original value from the original value.

`FILTER/SMO gordo flaco 1,1,0.5 YA CURSOR`
Smooth image 'gordo.bdf', average over a 3*3 neighbourhood and replace the original value by the average value only, if average value differs more than 0.5*original value from the original value but apply the averaging process only to subimages of 'gordo.bdf', which will be chosen via the cursor. It is assumed, that image 'gordo.bdf' is loaded into the image display!

`FILTER/SMOOTH gordo flaco 1,1,0.5 YA coords`
As above, but the subimage specifications are taken from the table 'coords.tbl'. This table could e.g. be created by using the command GET/CURSOR with the table option (and 2 cursors!)

`FILTER/SMO gordo flaco 3,0,0.  NA [@20,<:@50,>]`
Smooth image 'gordo.bdf' by averaging over a 7*1 neighbourhood; do not include the central value in the averaging and always replace the original value by the average value.
Apply the filter only to the subimage of frame 'gordo.bdf', which is defined by the coordinates given as parameter 5.

`FILTER/SMOOTH gordo flaco 3,2,22.3,24.6 NF ? nnn`
Apply boxcar filter only to pixels which have an intensity in the interval [22.3,24.6]. Do not include the central pixel in the calculation of the average value. Frame 'flaco.bdf' will have xdim = NPX-6 and ydim = NPY-4, with NPX, NPY the no. of pixels in x,y of the input frame 'gordo.bdf'.

# FIND/MINMAX

## FIND/MINMAX *core* 31-AUG-1984 KB

**Purpose:** Find minimum, maximum value of image frame and the corresponding pixel coordinates.

**Subject:** Minimum, maximum.

**Syntax:** `FIND/MINMAX frame`

`frame` name of image frame to work on

**See also:** STATISTICS/IMAGE, CUTS/IMAGE, FIND/PIXEL

**Note:** The image is scanned from first to last pixel. The coordinates of the first pixel found with minimum and maximum are displayed.
The minimum and maximum are also stored into real keyword OUTPUTR(1,2) and into real descriptor LHCUTS(3,4) of the frame.
The pixel coordinates are stored into integer key OUTPUTI(1-4) as xmin, ymin, xmax, ymax. The corresponding world coordinates are put into OUTPUTR(3,4).
The command STATISTICS/IMAGE provides (among other) the same information, but FIND/MINMAX is usually faster.

**Examples:** `FIND/MINMAX marano`
Display min, max and corresponding pixel coordinates of image 'marano.bdf'.

FIND/PIXEL *core* 23-JAN-1989 KB

**Purpose:** Find first or all pixel(s) with a value inside or outside the closed interval [low,high].

**Subject:** Minimum, maximum extremes.

**Syntax:** `FIND/PIXEL frame low,high [inout_flag] [first_flag] [table] [rowmax]`

| | |
|---|---|
| `frame` | name of frame to work on |
| `low,high` | low and high end of value-interval |
| `inout_flag` | IN or OUT, to indicate if we look inside or outside the interval [low,high]; defaulted to IN |
| `first_flag` | F(irst) or A(ll) if we want first pixel or all pixels in/outside the interval; defaulted to F |
| `table` | table name, only used with first_flag = ALL, to store all pixels in/outside the interval columns filled are :X_COORD, :Y_COORD, :VALUE; if omitted data are only displayed on the terminal |
| `rowmax` | max. no. of rows to fill in table given above; defaulted to 999 |

**See also:** STATISTICS/IMAGE, FIND/MINMAX

**Note:** For first_flag = F, the pixel value found is also stored into keyword OUTPUTR(1), corresponding pixel frame coordinates are stored in key OUTPUTI(1,2).
If a pixel with the desired value could not be found, OUTPUTI(1) and OUTPUTI(2) will contain 0,0 and OUTPUTR(1) is meaningless.
If first_flag = A, the no. of pixels found is stored in OUTPUTI(1)
The frame is scanned starting with lowest dimension first. All resulting data values are displayed on the terminal.

**Examples:** `FIND/PIXEL durazno -22.1,12.3 OUT`
Find first pixel in frame 'durazno.bdf' with a value which is less than -22.1 or greater than 12.3

`FIND/PIXEL durazno 97.3,10000000 ? ALL coords`
Find all pixels in frame 'durazno.bdf' with intensity >= 97.3 (assuming "normal" data values, i.e. less than 10 million ...) and store coordinates and values in table 'coords.tbl'.

# FIT/FLAT_SKY

## FIT/FLAT_SKY *core* 20-MAR-1984 KB

**Purpose:** Approximate background of an image by a surface created from a 2dim polynomial, using the least-squares method.

**Syntax:** `FIT/FLAT_SKY outframe = inframe [in_specs] [order] [back_surface]`

`FIT/FLAT_SKY inframe [in_specs] [order] [back_surface]`

| | |
|---|---|
| `outframe` | the computed background will be subtracted from the input image and stored in outframe |
| `inframe` | image where the reference points are taken from |
| `in_specs` | specifications for defining the background areas,<br>(a) CURSOR if the subimages are chosen via the cursor rectangle;<br>(b) name if the table 'name' holds the background area definitions in the columns :XSTART, :YSTART, XEND and :YEND;<br>defaulted to CURSOR |
| `order` | x-ord,y-ord, the orders in x and y of the approximating polynomial; defaulted to 1,1 |
| `back_surface` | name of frame to hold the background surface, if not given, the background is not saved |

**See also:** CONVERT/TABLE

**Note:** The "average" value in the background areas will be computed via a kappa-sigma clipping algorithm.

**Examples:** `FIT/FLAT ccd005 = ccd004 cursor 2,2 ccdback`
Use polynomial of order 2 in x and y to approximate the background of image 'ccd004.bdf'. The areas with the background are chosen interactively via the cursor rectangle. The approximated background is saved in image 'ccdback.bdf' .
Finally the background is subtracted from 'ccd004.bdf' and the result stored into frame 'ccd005.bdf' .

`FIT/FLAT ccd004 CURSOR 2,2 ccdback`
As above, but do not subtract background from 'ccd004.bdf' .

`FIT/FLAT ccd004 backgr 2,2 ccdback`
As above, but use the table 'backgr.tbl' to obtain the background areas of 'ccd004.bdf' .

FLIP/IMAGE *core* 13-JAN-1992 KB

**Purpose:** Flip an image around an axis inside the image.

**Syntax:** `FLIP/IMAGE frame [flag]`

| | |
|---|---|
| `frame` | image to be flipped (flipping is done in place...) |
| `flag` | X for flipping in x - i.e. around the center column;<br>Y for flipping in y - i.e. around the center row;<br>XY for flipping in x and y;<br>defaulted to X |

**See also:** TRANSPOSE/IMAGE, ROTATE/CLOCK, ROTATE/COUNTER_CLOCK
SHIFT/IMAGE, TRANSPOSE/CUBE

**Note:** The start and step descriptors are modified accordingly.

**Examples:** `FLIP/IMAGE matahari y`
Flip frame 'matahari.bdf' in y. The contents of the original frame are overwritten.

# GET/CURSOR

## GET/CURSOR *core* 05-JUL-1995 KB

**Purpose:** Read cursor coordinates from image display system.

**Syntax:** `GET/CURSOR [output] [option] [marker] [curs_specs] [zw_option]`

| | |
|---|---|
| `output` | specification for storage of world coordinates and pixel values, may be<br>table name (a),<br>descriptor name (b), specified as descr_name,DESCR<br>omitted (i.e. = ?) (c), if data is only displayed on the terminal;<br>defaulted to ? |
| `option` | append_flag for descriptors;<br>A => data values will be appended to the end of the descriptor, else data is written to beginning of the descriptor - this is the default;<br>append_flag+id_flag or no_flag for tables;<br>A => data values will be appended to the end of the table, else a new table is created - this is the default;<br>ID => for each cursor input the user is prompted for an identification (limited to 8 chars.);<br>NO or NO,start_no => enable automatic numbering of coordinate reading, starting at start_no;<br>this number is stored in a column labeled :No ;<br>defaulted to nothing of the above (see also the Notes) |
| `marker` | two-character flag for marking position in the overlay;<br>if first character = Y, a cross (rectangle) will be drawn in the overlay channel at the current cursor position(s);<br>if second character = Y, the identifier or sequence number will also be written to the overlay channel, only applicable for option ID and NO described above;<br>defaulted to YN |
| `curs_specs` | max number of coordinate readings and no. of cursors;<br>if set to 0,0 (or just 0) the cursor coordinates will be read out continuously and world coords. and intensity displayed;<br>defaulted to 9999,1 |
| `zw_option` | zoom_window_flag,zoom;<br>zoom_window_flag = W for zoom_window, N for none,<br>zoom = initial zoom factor;<br>only applicable to X11 window displays!<br>defaulted to N,4; |

**See also:** CREATE/CURSOR, SET/CURSOR, VIEW/IMAGE, EXTRACT/CURSOR, GET/GCURSOR

**Note:** If the output goes to a table the columns :X_coord,:Y_coord, :Value,:X_coordpix and :Y_coordpix will be created, if only one cursor involved.
If both cursors are involved, then for a rectangle the columns :Xstart,:Xend, :Ystart,:Yend,:Value1, :Value2, :Xstartpix, :Ystartpix, :Xendpix and :Yendpix will be created;
and for a circle the columns :X_coord,:Y_coord,:Value,:Radius1, :X_coordpix and :Y_coordpix will be created.
If option = NO or NO,number a column labeled :No is also created,
else a column :Ident is created which will contain an automatic identification (IDxyz) or if option = ID the user is prompted each time for an identification.
The append_flag and either id_ or no_flag can be combined via the '+' char., e.g. A+NO,2100 to append to the input table and fill the column :No with sequence numbers beginning at 2100.
If the output goes to a descriptor the world coords of x,y and the intensity are stored for a single cursor. For two cursors, the world coords of x,y and intensity of both cursors are stored, if a rectangle is used; and the world coords of x,y and intensity of the center pixel as weel as the radius are stored, if a circle is used.
– DeAnza Image Display –
The cursor box has to be set up with: Defined cursor(s) on, TRACK off, RATE on. Press the ENTER button to get the coordinates of the cursor(s) position. Set defined cursor(s) off + press ENTER to exit. However, when writing to a table with identifier column, TRACK has to be switched on! In this case, the typing of an identifier and RETURN on the keyboard will give the coordinates.
– XWindows –
Use the Mouse to move the cursor, the leftmost Mouse button or the RETURN key is the ENTER button, the right Mouse button serves as EXIT button.
This command also stores last obtained cursor pixels in keyword CURSOR(1,..,4), the last obtained coordinates in keyword OUTPUTR(10...) + total no. of coords. read in OUTPUTI(1).
If 'curs_specs' is NOT set to 'xxx,1' (which is the default!) a single cursor is used, irrespective of any previous SET/CURSOR command which might have initialized a cursor rectangle!
If 'curs_specs' is set to 0, i.e. continuous cursor read-out is done, always a single cursor is used.
Normally, the overlay channel is not cleared at startup of the command; if you want to clear first, set keyword AUX_MODE(9) = 1.
If we use the zoom_window mode, use first the cursor to choose a region in the main window. Press ENTER to copy the indicated rectangle to the zoom window. Move the cursor into this window and get the cursor values. To choose another region, press EXIT in the zoom window, move the cursor back to the main window and choose another region. To exit, press EXIT in the main window.
Also options exist for changing the zoom factor, using color LUTs, different ITTs and scrolling (like in VIEW/IMAGE), which can be activated by typing a single key (with the cursor in the main display window), type 'h' for help on that.
If a cursor window exists (cf. CREATE/CURSOR) an area of 9x9 pixels around the cursor #0 (i.e. the mouse) is shown in the cursor window zoomed up by a factor of 20.

**Examples:** `GET/CURSOR`
Display cursor data on terminal only.

`GET/CURSOR coords,descr ? N`
Also store world coordinates and intensity into real descriptor COORDS of displayed frame. Inhibit drawing of cross at cursor position.

`GET/CURSOR coords`
Create new table 'coords.tbl' and store data into that table.

`GET/CURSOR coords A`
Append data to existing table 'coords.tbl'.

# GET/CURSOR

`GET/CURSOR coords ID`
Create new table, do not use automatic, sequential identifiers, but get them from the user.

`GET/CURSOR coords A+ID`
As above, but append to existing tabel 'coords.tbl' .

`GET/CURSOR coords A+ID YY`
Also draw the identifier into the overlay channel.

`GET/CURSOR p4=0`
When the cursor is inside the image display window read out the cursor position continuously and display data on terminal.

`GET/CURSOR p5=w`
Use auxiliary zoom window to get the cursor coordinates. If the zoom window does not exist yet, it will be created with half the size in x and y of main display window.

`GET/CURSOR p4=0 p5=w`
With the cursor choose a region inside the main image display window. Then inside the zoom window read out the cursor position continuously and display data on terminal.

GET/GCURSOR *core* 13-JAN-1990 RHW

**Purpose:** Read and store cursor coordinates from the graphics display.

**Subject:** Position, graphics.

**Syntax:** `GET/GCURSOR [output_spec] [app_flag] [max]`

| | |
|---|---|
| `output_spec` | output specification to receive coordinates and/or data values. The output can be stored in a table or a descriptor. To store the output in a table the specification should be 'table_name,TABLE'; if the output is to be stored in a descriptor the specification should be 'descr_name,DESCR'. The latter is only possible if the data plotted originate from a frame or a table. |
| `app_flag` | 'A' indicates that data values will be appended to the table or descriptor used as output; default the descriptor or table is created or overwritten. |
| `max` | optional limit to the number of coordinate pairs read; default = 99999 |

**Output:** Table or descriptor containing positional information.

**Note:** The command can be used after each main plot and overplot command (to display descriptor, keyword, table or image data, or to plot a set of axes). By default, cursor values are only displayed on the terminal.
IMPORTANT: The position(s) retrieved by the cursor refer to the last plotted data. Hence, after a plot and subsequent overplot command(s) the coordinates refer to the data plotted with the last overplot command.
The command enables the graphic cursor and reads the screen coordinates. If the graphics cursor is used on table, keyword, or descriptor data, the screen (world) coordinates are read and stored in the output table in the columns X_AXIS and Y_AXIS. In case of image data the command converts the screen coordinates to pixel and world coordinates of the nearest image pixel. These numbers are stored in the columns labelled X_AXIS, Y_AXIS, LINE_NO, PIXEL_NO, X_COORD, Y_COORD. The pixel intensity is stored in the column VALUE.
For XWindows use the mouse to move the cursor, the leftmost mouse button or the RETURN key is the ENTER button, the second left mouse button serves as EXIT button. The cursor cross may not be visible at first. Move the mouse to the lower left corner of the graphics window to grab the cursor.
For graphic terminals pressing any key of the graphics keyboard will give the coordinates of the cursor(s) position; the space bar is used to exit from the cursor mode.

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, SET/GCURSOR

**Examples:** `GET/GCURS`
display cursor data on terminal only

`GET/GCURS COORDS,DES`
as before but store world coords. and intensity into real descriptor 'COORDS' of the frame or table for which the data was plotted

`GET/GCURS coords A`
as above but append the table 'coords' with these data

# GET/IMAGE

## GET/IMAGE *core* 17-OCT-1983 KB

**Purpose:** Read image from displayed image channel

**Syntax:** `GET/IMAGE frame [input_source] [ITT_flag]`

| | |
|---|---|
| `frame` | name of image frame, where the pixel data will be stored |
| `input_source` | channel no. (a), CURSOR (b) or CURSOR,xpix,ypix (c);<br>for (a) the whole image in given channel is read out;<br>for (b) a cursor rectangle will determine the size of a subimage to be read out;<br>for (c) a single cursor is used to define the center of a rectangle of size 'xpix,ypix' ;<br>defaulted to currently displayed channel, i.e. option (a) |
| `ITT_flag` | if set to ITT, pixels are also ITT-mapped, else not;<br>defaulted to NOITT |

**See also:** COPY/DISPLAY, EXTRACT/CURSOR, GET/CURSOR

**Note:** Only the part of the cursor rectangle which covers the displayed image is used. Therefore, you may get a smaller image than expected, if the displayed image does not fill all the display memory and your cursor rectangle is not completely inside the displayed image.
For a description of how to move and modify the cursor rectangle use HELP [ImageDisplay].

**Examples:**

`GET/IMA tecate 0`
Store complete image of channel 0 into frame tecate.bdf .

`GET/IMA tecate 0 ITT`
As above, but contents of channel are mapped via loaded ITT.

`GET/IMA bohemia CURSOR`
Store image defined by cursor rectangle into frame bohemia.bdf .

`GET/IMA dosequis CURSOR,200,220`
Store image of size 200*220 pixels into frame dosequis.bdf. The center is determined via single cursor cross.

`GET/IMA dosequis CURSOR,200,220 ITT`
As above, but contents of channel are mapped via loaded ITT.

GET/ITT *core* 23-JUL-1990 KB

**Purpose:** Read currently active ITT from image display.

**Syntax:** `GET/ITT out_specs [chanl] [sect]`

| | |
|---|---|
| `out_specs` | table or image,descr<br>(a) table - table name where the currently active ITT will be stored<br>(b) image,descr - name of frame and descriptor where the currently active ITT will be stored |
| `chanl` | channel using the ITT, defaulted to active channel |
| `sect` | section no. of ITT table, default section is 0. |

**See also:** GET/LUT, LOAD/LUT, LOAD/ITT, TUTORIAL/ITT, MODIFY/ITT

**Note:** The ITT file will be stored in your current directory. For option (b), the image has to exist in your current directory (the data directory for MIDAS).

**Examples:** `GET/ITT gordo,MIDAS_ITT`
Get current ITT and store it in descriptor MIDAS_ITT of frame 'gordo.bdf' in your current directory.

`GET/ITT beautiful`
Get current ITT and store it in table 'beautiful.itt' in the directory MID_WORK.

# GET/LUT

GET/LUT *core* 10-JAN-1995 KB

**Purpose:** Read currently active LUT from image display.

**Syntax:** `GET/LUT out_specs [get_specs] [ITT] [format] [range]`

| | |
|---|---|
| `out_specs` | table or image,descr<br>(a) table - table name where the currently active LUT will be stored<br>(b) image,descr - name of frame and descriptor where the currently active LUT will be stored<br>(c) ASCII file name - each line will contain the red, green and blue LUT component separated by a blank |
| `get_specs` | section, no. of values - defaulted to 0,256<br>this parameter is only applicable for DeAnza displays (!) |
| `ITT` | optional ITT_flag, if given, the LUT values are mapped via the current ITT; |
| `format` | TABLE or ASCII, for output to Midas binary table or ASCII;<br>defaulted to TABLE |
| `range` | interval for LUT values if output to ASCII file,<br>either 0,1 or 0,255 is supported; defaulted to 0,1 |

**See also:** GET/ITT, LOAD/LUT, LOAD/ITT, TUTORIAL/LUT, MODIFY/LUT

**Note:** The LUT file will be stored in your current working directory. For option (b), the image has to be resident in your current directory (the data directory for MIDAS).

**Examples:** `GET/LUT gordo,MIDAS_LUT`
Get current LUT and store it in descriptor MIDAS_LUT of frame 'gordo.bdf' in your current directory.

`GET/LUT beautiful`
Get current LUT and store it in table 'beautiful.lut' in the current working directory.

`GET/LUT doubtful.mmm ? ? ASCII 0,255`
Get current LUT and store it in the ASCII file 'doubtful.mmm' in the current working directory. The LUT values are integers in the interval [0,255].

## GROW/IMAGE *core* 22-FEB-1993 KB

**Purpose:** Make a 2 dimensional file from a single scan line, repeating that line as a line or column a given number of times.

**Syntax:** `GROW/IMAGE out = in [start,step,no] [lincol_specs] [lincol_flag]`

| | |
|---|---|
| `out` | resulting 2 dimensional frame |
| `in` | 1- or 2-dim frame from which the input line is to be taken |
| `start,step,no` | the start and step size in the y- or x-dimension (depends upon 'lincol_flag') of result frame and the no. of repetitions, i.e. new y- or x-dimension; defaulted to 0.,1.,NPIX(1) of 'in' |
| `lincol_specs` | 'coord,LINE' or 'coord,COLUMN' to indicate for a 2-dim input frame which line or column of the frame is to be used as input scan line; the y- or x-coordinate of the line or column is defined in the usual MIDAS syntax in pixel or world coordinates; defaulted to '<,LINE', which indicates the first line |
| `lincol_flag` | L or C for line- or columnwise repetition of the input line/column; defaulted to L (also everything else but C is taken as L) |

**See also:** EXTRACT/IMAGE, INSERT/IMAGE, ROTATE/1DIM

**Note:** In general, MIDAS works more efficiently on line oriented frames, i.e. frames where the no. of pixels in a line (row) is greater than or equal to the no. of columns.

**Examples:** `GROW/IMA mucho = nada 1.,1.,10`
Create 2-dim frame mucho.bdf by replicating 10 times the first line of frame nada.bdf. Start in y is 1.0 and y-stepsize is also 1.0.

`GROW/IMA rio.grande = colorado ? @22 col`
Create 2-dim frame rio.grande by replicating the 22. line of frame colorado.bdf as columns. Start in x is 0.0 and x-stepsize is 1.0; the no. of columns (y-dim) of rio.grande is NPIX(1) (i.e. x-dim) of colorado.bdf.

## HELP/APPLIC *core* 13-APR-1984 KB

**Purpose:** Display header information of application procedures which are stored in APP_PROC:

**Syntax:** `HELP/APPLIC [proc]`

| | |
|---|---|
| `proc` | name of procedure without type, ".prg" is appended; if omitted a menu of all available application procedures is displayed, this is the default |

**See also:** description of the help facility in chapter 3 of the MIDAS Users Guide, volume A

**Note:** These application procedures provide you with a set of additional MIDAS commands. These functions are less often used, therefore we did not create command names for them.
One of these application procedures, "prhelp.prg" can be used to also print the output of the HELP/APPLIC command.

**Examples:** `HELP/APP slicube`
Display help info about the parameters used with the procedure 'slicube.prg' .

# HELP/CL

## HELP/CL *core* 12-DEC-1984 KB

**Purpose:** Display help for commands only used in MIDAS procedures.

**Syntax:** `HELP/CL [command]`

| | |
|---|---|
| `command` | command about which info is required;<br>if omitted, a menu of all available commands for MIDAS procedures is displayed;<br>this is the default |

**See also:** Detailed description of the help facility and the MIDAS procedures in chapter 3 of the MIDAS Users Guide, Volume A

**Note:** None.

**Examples:** `HELP/CL IF`
Will display help for the IF command in MIDAS procedures.

## HELP/CONTRIB *core* 11-JAN-1995 KB

**Purpose:** Display header information of application procedures which are stored in CON_PROC:

**Syntax:** `HELP/CONTRIB [proc]`

| | |
|---|---|
| `proc` | name of procedure without type, ".prg" is appended;<br>if omitted a menu of all available application procedures in the Midas 'contrib' area is displayed;<br>this is the default |

**See also:** HELP/APPLIC
description of the help facility in chapter 3 of the MIDAS Users Guide, volume A

**Note:** These application procedures provide you with a set of additional Midas commands. They are executed via "@c procedure". These procedures involve contributed software (as well as public domain software). They provide functionality which is (usually) less often used, therefore we did not create Midas command names for them.

**Examples:** `HELP/CONTRIB lutedit`
Display help info about the Motif based LUT editor, launched via "@c lutedit".

## HELP *core* 8-AUG-1989 KB

**Purpose:** Display help information.

**Syntax:** `HELP [help_topic]`

`help_topic` command (a) or command/qualifier (b)
(a), all command/qualifier combinations of command are displayed
(b), complete information about the specific command/qualifier is displayed
if help_topic is omitted, a general help menu is displayed;
this is the default

**Note:** Commands and qualifiers may be truncated to the significant number of characters.
To get short info about a specific command, append '??' as first parameter to the command.
To get all commands beginning with same string, append '?' to the string.

**See also:** detailed description of the help facility in chapter 3 of the MIDAS Users Guide, volume A
PRINT/HELP, HELP/APPLIC, HELP/CONTRIB, HELP/CL
HELP/KEYWORD, HELP/DESCR

**Examples:** `HELP LOAD`
Will display all different options (qualifiers) of command LOAD

`HELP LOAD/IMAGE`
Will display complete help info about command LOAD/IMAGE

`LOAD/IMAGE ??`
Will display short help info about LOAD/IMAGE

`LO?`
Will display all commands beginning with the string LO

`HELP BYE/`
Will display info about command BYE. Note the way we indicate a command/qualifier option, if no qualifier exists.

## HELP/KEYWORD *core* 20-MAY-1987 KB

**Purpose:** Explain contents of given keyword.

**Syntax:** `HELP/KEYWORD key`

`key` name of keyword we want to get a description of;
the key has to be a system or fixed key, i.e. a key stored in the initial keyword file, MID_MONIT:syskeys.dat

**See also:** Detailed description of the help facility in chapter 3 of the MIDAS Users Guide, Volume A

**Note:** None

**Examples:** `HELP/KEY AUX_MODE`
Explain the contents of keyword AUX_MODE.

# HELP/QUALIFIER

## HELP/QUALIFIER *core* 19-OCT-1983 KB

**Purpose:** Display all MIDAS commands with given qualifier.

**Syntax:** `HELP/QUALIF [qualif]`

`qualif` name of qualifier;
defaulted to wildcard (matches all patterns) qualifier ( = ...)

**See also:** Detailed description of the help facility in chapter 3 of the MIDAS Users Guide, Volume A

**Note:** If qualifier is not unique all matching qualifiers are used.
It's good practice to look also for all the commands with a wildcard qualifier ("...") because they might accept the given qualifier as well.

**Examples:** `HELP/QUALIF ICAT`
Display all MIDAS commands with qualifier ICAT. Also the commands with a wildcard qualifier are listed because they might (!) accept this qualifier as well.

`HELP/QUALIF ...`
Display all MIDAS commands with a wildcard qualifier, which is indicated via "..." in MIDAS.

## HELP/SUBJECT *core* 27-MAR-1987 KB

**Purpose:** Display information related to given topic.

**Syntax:** `HELP/SUBJECT [topic]`

`topic` any topic about which you need information, try it...
if omitted, all available topics are shown; this is the default

**See also:** description of the help facility in chapter 3 of the MIDAS Users Guide, volume A
HELP, HELP/QUALIF

**Note:** None.

**Examples:** `HELP/SUBJECT`
Get a list of all available topics.

`HELP/SUBJECT zooming`
Get a list of MIDAS commands related to zooming.

## INDISK/ASCII *core* 02-MAR-1995 KB

**Purpose:** Read ASCII file from disk + convert to Midas image

**Syntax:** `INDISK/ASCII in_file [out_file] [npix_string]`

| | |
|---|---|
| `in_file` | name of input ASCII file |
| `out_file` | name of Midas image (max. 3 dimensions); defaulted to toto.bdf |
| `npix_string` | optional string with naxis,npix(1),npix(2),npix(3); if not given, a 1-dim image with the no. of pixels read from the ASCII file is created; that is the default |

**See also:** INDISK/FITS, INTAPE/FITS, CREATE/IMAGE

**Note:** If the ASCII file has less pixels than required (possible, if npix_string is specified) the remaining image pixels are set to the contents of keyword NULL(1), the user defined null value.

**Examples:** `INDISK/ASCII zapato.asc bota`
Read ASCII file 'zapato.asc' and convert it to the 1-dim image 'bota,bdf'. NPIX(1) of 'bota.bdf' will be set to the no. of pixels stored in 'zapato.asc'.

`INDISK/ASCII ascii.data ? 2,200,400`
Read ASCII file 'ascii.data' and convert it to the 2-dim image 'toto,bdf', NPIX(1,2) of 'toto.bdf' is set to 200,400. If 'ascii.data' does not contain 80000 pixels, the remaining pixels of 'toto.bdf' are set to the contents of keyword NULL(1).

# INDISK/FITS

## INDISK/FITS *core* 17-DEC-1993 KB

**Purpose:** Read FITS/IHAP files from disk.

**Syntax:** `INDISK/FITS in_files [out_files] [option]`

| | |
|---|---|
| `in_files` | filename(s) of FITS files separated by a comma (a)<br>or name of ASCII file (indicated by filetype '.cat') which contains a FITS filename on each line (b);<br>also wildcard characters are supported. |
| `out_files` | corresponding MIDAS filenames separated by a comma for (a)<br>or name of ASCII file depending upon if 'in_files' is list of filenames or ASCII catalog ((b) above) |
| `option` | O(riginal) if the result files should be named according to the FITS keyword FILENAME, i.e. we also execute RESTORE/NAME<br>or N(one);<br>defaulted to N |

**See also:** INTAPE/FITS, OUTTAPE/FITS, RESTORE/NAME, INDISK/ASCII

**Note:** If the output file names are omitted or not as many as the input file names, the names 'toto0001.bdf', 'toto0002.bdf', ... are used.
The command works for images, tables and fitfiles.
Also the wildcard char. (*) is supported in the innput file string.

**Examples:** `INDISK/FITS xyz`
Convert FITS file 'xyz' to 'toto0001.bdf' or 'toto0001.tbl' or 'toto0001.fit' depending upon if 'xyz' was an image, table or fit file.

`INDISK/FITS tst0003.mt prado.midas`
Convert FITS file 'tst0003.mt' to 'prado.midas'

`INDISK/FITS paradis.cat paradis.out`
Read ASCII file 'paradis.cat' and convert the FITS files with names in that file to MIDAS files with the corresponding names in file 'paradis.out'

`INDISK/FITS luminy*,canebiere.fits panier.dat`
Convert all FITS files matching the pattern 'luminy*' as well as file 'canebiere.fits' to MIDAS files with names according to the entries in ASCII file 'panier.dat'

## INFO/DESCR *core* 11-MAY-1992 KB

**Purpose:** Get type, no. of elements and bytes per element of descriptor.

**Syntax:** `INFO/DESCR frame descr`

| | |
|---|---|
| `frame` | name of image or table or fit file<br>file type defaulted to '.bdf' |
| `descr` | descriptor name |

**See also:** SHOW/DESCR, READ/DESCR, INFO/IMAGE, COMPUTE/KEYWORD

**Note:** This command is especially useful inside MIDAS procedures. No output is generated, only the integer keyword MID$INFO is filled as follows:
If the frame could not be opened MID$INFO(1) = -1.
IF the descriptor could not be found, MID$INFO(1) = 0.
Otherwise MID$INFO(1) contains the type of the descriptor,
1 = integer, 2 = real, 3 = character, 4 = double precision.
MID$INFO(2) holds the no. of elements of the descriptor and MID$INFO(3) has the no. of bytes per element.
The function M$EXISTD of the command COMPUTE/KEYWORD provides similar but less detailed information.

**Examples:** `INFO/DESC durazno start`
If frame 'durazno.bdf' is a valid MIDAS image, MID$INFO(1) = 4, MID$INFO(2) >= 1 and MID$INFO(3) = 8.

`INFO/DESC pera.tbl tblcontr`
If table 'pera.tbl' is a valid MIDAS table, MID$INFO(1) = 1, MID$INFO(2) = 10 and MID$INFO(3) = 4.

## INFO/IMAGE *core* 12-JUL-1995 KB

**Purpose:** Get internal info of image frame.

**Syntax:** `INFO/IMAGE frame`

| | |
|---|---|
| `frame` | name of image, file-type defaulted to '.bdf' |

**See also:** READ/DESCR, INFO/DESCR

**Note:** This command displays in a user readable format the values of the internal File Control Block (FCB) of a binary Midas image. The FCB (512 bytes) is the very first block of every Midas file.
Also the integer keyword MID$INFO is filled as follows:
If the image frame could not be opened MID$INFO(1) = -1, else
MID$INFO(1) contains the type of the image data, D_R4_FORMAT (10), D_I1_FORMAT(1), ...
MID$INFO(2) holds the no. of pixels of the image and MID$INFO(3) has the no. of bytes per pixel.

**Examples:** `INFO/IMAGE durazno`
Display all relevant FCB values of Midas image 'durazno.bdf' and fill keyword MID$INFO.

# INFO/SETUP

## INFO/SETUP *core* 27-JAN-1994 KB

**Purpose:** Display all the information about a Setup.

**Syntax:** `INFO/SETUP [setup]`

| | |
|---|---|
| `setup` | name of a Setup<br>if omitted, all currently implemented Setups are listed |

**See also:** INITIALIZE/SETUP, READ/SETUP, WRITE/SETUP

**Note:** Setups are a collection of variables (keywords) which have to be set up before certain, more complex commands can be executed.

**Examples:** `INFO/SETUP catalog`
Display information about the Setup 'catalog'.

`INFO/SETUP`
List all currently implemented Setups.

## INITIALIZE/DISPLAY *core* 24-Jan-1994 KB

**Purpose:** Initialize the image display.

**Syntax:** `INITIALIZE/DISPLAY [noLUT,LUTsz] [ownLUT] [M_unit] [fonts]`

| | |
|---|---|
| `noLUT,LUTsz` | the no. of possible LUTs and the no. of entries in each LUT; defaulted to 1,220 |
| `ownLUT` | indicates if we have our own LUT<br>= -1 (companion LUT); = 0 (no own LUT); = 1 (own LUT);<br>defaulted to 1 |
| `M_unit` | unit of Midas session which "owns" the colors of the display, this parameter is only needed for ownLUT = -1 |
| `fonts` | font no.s for normal, large and very large fonts;<br>the font numbers belonging to a given font name may be found in the file 'MID_SYSTAB:x11fonts.dat';<br>defaulted to -1,-1,-1 which means use default font names |

**See also:** CREATE/DISPLAY, DELETE/DISPLAY

**Note:** The parameters described above are only used for Xwindow systems!
If 'ownLUT' = -1, i.e. in companion mode, the first two parameters are not used, the values are taken from the companion MIDAS, instead.
For DeAnza image displays INITIALIZE/DISPLAY does:
The image memory is cleared, the last available channel is initialized as graphics channel and default tables are loaded.

**Examples:** `INITIALIZE/DISPLAY`
Initalize an image display; if it's an Xwindow station use the default values for the parameters.

`INITIALIZE/DISPLAY 1,200`
Indicate that you have 200 color cells available for the LUT and only one LUT.

`INITIALIZE/DISPLAY 4,256`
Indicate that you have 4 LUTs of 256 color cells each available. This is e.g. true on a (now defunct) Stellar workstation.

`INITIALIZE/DISPLAY ? -1 34`
Indicate that you use the colors from Midas session with unit 34.

`INITIALIZE/DISPLAY p4=24,176,1022`
Set up display to use the fonts which are listed as no. 24, no. 176 and no. 1022 in the file 'MID_SYSTAB:x11fonts.dat'.

# INITIALIZE/SETUP

## INITIALIZE/SETUP *core* 27-JAN-199- KB

**Purpose:** Initialize the variables of a Setup.

**Syntax:** `INITIALIZE/SETUP setup`

`setup` name of a Setup

**See also:** INFO/SETUP, READ/SETUP, WRITE/SETUP

**Note:** Setups are a collection of variables (keywords) which have to be set up before certain, more complex commands can be executed.
To get a list of all currently implemented Setups use the command INFO/SETUP.
Use "READ/SETUP setup" to see the default values for the variables of a specific Setup 'setup' after having executed the command "INIT/SETUP setup".

**Examples:** `INIT/SETUP catalog`
Set the variables of the Setup 'catalog' to their default values. With "READ/SETUP catalog" these variables may be displayed.

INSERT/IMAGE *core* 14-MAY-1991 KB

**Purpose:** Insert a subframe into another frame.

**Syntax:** `INSERT/IMAGE subframe modframe [startx,y,z]`

| | |
|---|---|
| `subframe` | name of image frame which will be inserted in the other |
| `modframe` | name of image frame into which 'subframe' will be inserted;<br>parts or all of modframe will be overwritten with the pixels of subframe, depending upon the start values |
| `startx,y,z` | optional start coordinates for subframe within modframe;<br>if omitted the start coords. will be taken from the descriptor START of the subframe |

**See also:** EXTRACT/IMAGE

**Note:** startx,starty,startz can be in any of the following formats:
- any number, to indicate real world coordinates;
- an integer number preceded by @, to indicate a pixel number;
- the symbols "<" and ">", to indicate start or end pixel

the startvalues may also be negative, i.e. "before" the start of the modframe, so that only the overlapping part of the subframe is put into the modframe
since @1 indicates first pixel, @0 means 1 pixel outside and @-1 actually puts the begin of the insertion 2 pixels outside modframe!

**Examples:** `INSERT/IMA sun galaxy`
insert frame sun.bdf into galaxy.bdf, the position of sun within galaxy will be determined by descriptor START of sun.bdf

`INSERT/IMA sun galaxy @22,@100`
insertion of the data of sun.bdf will begin at the 22th x-pixel and the 100th y-pixel of frame galaxy.bdf (assuming, that both frames are 2-dimensional frames)

`INSERT/IMA sun galaxy @-1,@-1`
insert sun.bdf into galaxy.bdf, sun will begin two pixels and two lines outside of modframe, therefore only from pixel 3 and line 3 on are data overlapping and thus inserted into galaxy.bdf

# INTAPE/FITS

## INTAPE/FITS *core* 14-JAN-1994 PJG

**Purpose:** Read image frames from magtape or disk in FITS/IHAP format.

**Syntax:** `INTAPE/FITS file_specs file_id device [flags]`

| | |
|---|---|
| `files_specs` | numbers of frames as stored on tape in ascending order, e.g. 2,17,22-33,44 specifies frames 2,17,22 till 33,44 |
| `file_id` | file identification (max. 4 chars), to this string the file no. on tape is appended, e.g. file no. 17 will be stored in file_id0017.bdf |
| `device` | logical tape unit (e.g. TAPE1), physical tape unit (e.g. /dev/nrst8 or host:/dev/nrst1) or prefix of file name on disk. For disk files the extension .mt is assumed. It is also possible to specify a full file name with extension in which case the data are taken from it. |
| `flag` | 3-character flag: list, create, history<br>list flag: F(ull), S(hort), N(one)<br>create flag: N(o create), O(original format), F(loating point format)<br>history flag: H(istory stored) or N(o)<br>Default flags are SOH |

**See also:** OUTTAPE/FITS, INDISK/FITS, CREATE/xCAT, SET/xCAT

**Note:** The input tape must be loaded on a tape unit before executing the command. All terminal output is stored in the MIDAS logfile as well.
To access remote tape drives, the MIDAS tapeserver demon must be installed and run on the host in question.
A listing of file headers can be made by using the flags FNN
For FITS format, only the descriptor elements corresponding to FITS keywords are created. Thus, some elements (e.g. in O_TIME) may not be initiated.

**Examples:** `INTAPE/FITS 1-15 ccd TAPE0`
–read image frames no. 1 through 15 on the tape mounted on unit TAPE0 and store them into frames ccd0001 to ccd0015

`INTAPE/FITS * x /dev/nrst8 FNN`
–list headers of all files from magtape mounted on the local tape unit /dev/nrst8.

`INTAPE/FITS * x obs:/dev/nrst8 FNN`
–list headers of all files from magtape mounted on the tape unit /dev/nrst8 on host 'obs'. NOTE: MIDAS tape setver must run on this host.

`INTAPE/FITS 1-2 data image`
–read from file image0001.mt and image0002.mt creating the frames data0001.bdf and data0002.bdf.

`INTAPE/FITS 1 gal galaxy.fits`
–read from file galaxy.fits and create the frame gal0001.bdf.

## INTERPOLATE/II *core* 29-OCT-1985 JDP

**Purpose:** Spline interpolation of 1D data. Image to Image transformation.

**Syntax:** `INTERPOLATE/II outima inima refima [s] [degree]`

| | |
|---|---|
| `outima` | name of output image |
| `inima` | name of input image |
| `refima` | reference image |
| `s` | smoothing parameter (default 1.) |
| `degree` | degree of spline (default 3, cubic splines) |

**Note:** The parameter s controls the degree of smoothing. This parameter has to be chosen carefully: too small s-values will result in an overfitting, too large s-values will produce an underfitting of the data. For s very large it returns the least-squares polynomial fit.

The number of spline knots and their positions are determined automatically, taking into account the specific behaviour of the function underlying the data.

Ref.: P. Dierckxx, 1982, Computer Graphics and Image Processing, vol. 20, 171-184.

********** Experimental version **********

**Examples:** `INTERPOLATE/II OUTPUT INPUT REFIMA 1. 3`

## INTERPOLATE/IT *core* 29-MAY-1987 JDP

**Purpose:** Spline interpolation of 1D data. Image to Table transformation.

**Syntax:** `INTERPOLATE/IT outtab i,d inima [s] [degree]`

| | |
|---|---|
| `outtab` | name of output table |
| `i` | column reference for independent variable |
| `d` | column reference for output variable. If the column does not exist it is created |
| `inima` | name of input image |
| `s` | smoothness parameter (default 1.) |
| `degree` | degree of spline (default 3, cubic splines) |

**Note:** The parameter s controls the degree of smoothing. This parameter has to be chosen carefully: too small s-values will result in an overfitting, too large s-values will produce an underfitting of the data. For s very large it returns the least-squares polynomial fit.

The number of spline knots and their positions are determined automatically, taking into account the specific behaviour of the function underlying the data.

Ref.: P. Dierckxx, 1982, Computer Graphics and Image Processing, vol. 20, 171-184.

********** Experimental version **********

**Examples:** `INTERPOLATE/IT TABLE :X,:Y IMAGE 1. 3`

# INTERPOLATE/TI

## INTERPOLATE/TI *core* 29-MAY-1987 JDP

**Purpose:** Spline interpolation of 1D data. Table to Image transformation.

**Syntax:** `INTERPOLATE/TI outima intab i,d refima [s] [degree]`

| | |
|---|---|
| `outima` | name of output image |
| `intab` | name of input table |
| `refima` | reference image |
| `i` | column reference for independent variable |
| `d` | column reference for dependent variable |
| `s` | smoothness parameter (default 1.) |
| `degree` | degree of spline (default 3, cubic splines) |

**See also:** REBIN/TI, CONVERT/TABLE

**Note:** Values in the column used as independent variable must be monotonically increasing or decreasing.

The parameter s controls the degree of smoothing. This parameter has to be chosen carefully: too small s-values will result in an overfitting, too large s-values will produce an underfitting of the data. For s very large it returns the least-squares polynomial fit.

The number of spline knots and their positions are determined automatically, taking into account the specific behaviour of the function underlying the data.
Ref.: P. Dierckxx, 1982, Computer Graphics and Image Processing, vol. 20, 171-184.

Examples:

```
INTERPOLATE/TI IMAGE TABLE :X,:Y REFIMA 1.  3
```

## INTERPOLATE/TT *core* 29-MAY-1987 JDP

**Purpose:** Spline interpolation of 1D data. Table to Table transformation.

**Syntax:** `INTERPOLATE/TT outtab i,d intab i,d [s] [degree]`

| | |
|---|---|
| `outtab` | name of output table |
| `intab` | name of input table |
| `i` | column reference for independent variable |
| `d` | column reference for dependent variable. If the output column does not exist it is created |
| `s` | smoothness parameter (default 1.) |
| `degree` | degree of spline (default 3, cubic splines) |

**Note:** Values in the column used as independent variable must be monotonically increasing or decreasing.

The parameter s controls the degree of smoothing. This parameter has to be chosen carefully: too small s-values will result in an overfitting, too large s-values will produce an underfitting of the data. For s very large it returns the least-squares polynomial fit.

The number of spline knots and their positions are determined automatically, taking into account the specific behaviour of the function underlying the data.
Ref.: P. Dierckxx, 1982, Computer Graphics and Image Processing, vol. 20, 171-184.

********** Experimental version **********

**Examples:** `INTERPOLATE/TT OUTABLE :X,:Y INTAB :X,:Y 1. 3`

## ITF/IMAGE *core* 02-JUNE-1986 MR

**Purpose:** Apply Intensive Transformation Function correction to frames using a calibration table.

**Subject:** Intensity transformation, calibration.

**Syntax:** `ITF/IMAGE inframe table coli,colo scal outframe`

| | |
|---|---|
| `inframe` | name of input data frame to be transformed |
| `table` | name of calibration table containing the intensity transformation |
| `coli` | column with input intensities |
| `colo` | column with output intensities |
| `scal` | scaling for input column of table |
| `outframe` | name of output frame |

**Output:** outframe

**Note:** The output values are linearly interpolated from the intensities in the table.

**Examples:** `ITF/IMAGE INPUT ITF #1,#2 1 OUT`

# JOIN/TABLE

## JOIN/TABLE *core* 09-SEP-1992 MP

**Purpose:** Find common objects in two tables by comparing one or two attributes

**Syntax:** `JOIN/TABLE intab1 :X1,[:Y1] intab2 :X2,[:Y2] outtable [tolX,tolY]`

| | |
|---|---|
| `intab1` | first input table |
| `:X1,:Y1` | columns of the table intable1 to be looked at |
| `intab2` | second input table |
| `:X2,:Y2` | columns of the table intable2 to be looked at |
| `outtable` | output table |
| `tolX,tolY` | tolerance for the error. defaulted to 0,0<br>only valid for numerical columns |

**Note:** The command joins table files, that is it finds common objects in two tables by comparing one or two attributes (columns of the table) of the objects (rows of the table) from both files, given an uncertainty for each attribute. In the exact match (uncertainty equal to 0) objects with identical attributes are identified as being the same object. In the approximate match, the attributes have to be the same within a certain tolerance. Only exact match can be performed on character strings columns. The (numerical) columns which are compared don't need to be of the same type: the match will be done in double precision. The output table will contain as many rows as the number of common objects which are found. It will contain all the columns of the first table as well as the ones from the second table. The columns will get the labels laborig_n where laborig is the label of the column in the input table and n the number of the input table.
The biggest of the two input tables should be given in the command line as second input table (intab2)

**Examples:** `JOIN/TABLE zcat :RA,:DEC rc3 :RA,:DEC zcat_rc3 0.01,0.01`
Find the common objects from the tables 'rc3.tbl' and 'zcat.tbl' by comparing their columns RA and DEC. Objects having similar coordinates within a error box or 0.01*0.01 will be identified as common and all their attributes will be copied into the table 'zcat_rc3.tbl'.

## LABEL/DISPLAY *core* 16-NOV-1994 KB

**Purpose:** Write a character string into the alpha-numerics or overlay channel of the image display.

**Syntax:** `LABEL/DISPLAY labl [position] [mode] [option] [size] [key_flag]`

| | |
|---|---|
| `labl` | string of max. 80 characters, if the string contains blanks, the whole string has to be enclosed within double quotes ("); or name of char. keyword containing the string(s) (see 'key_flag') |
| `position` | line and column number, where to write the string or CURSOR, if line and column will be picked via cursor or name of integer keyword containing the position(s); defaulted to CURSOR |
| `mode` | A(lpha) or O(verlay), for writing into the alphanumeric memory or overlay channel; defaulted to O for XWindow displays, to A for all others; invalid input defaults also to O |
| `option` | for X11 displays - color of label, if mode = OV: defaulted to white; for DeAnza displays - alpha-submode, if mode = A: 0 = white on black,1 = yellow on black, 2 = black on white, 3 = black on yellow, defaulted to 0; or angle in degrees, defaulted to 0, if mode = OV; |
| `size` | 0, 1, 2 for Normal, Large or VeryLarge; default = 0 Only valid for X11 devices! |
| `key_flag` | K(ey) or N(okey) to indicate that 'labl' and 'position' hold names of keywords which contain the labels and positions; up to 20 labels with a total max. of 200 chars. may be stored in the character keyword, each label is terminated by a single tilde char. (" "). The last label is terminated by two consecutive tildes (" "). defaulted to N |

**See also:** CLEAR/DISPLAY, CLEAR/ALPHA

**Note:** The alpha-numerics memory is structured as an array of 25 lines (0 - 24) and 80 columns (0 - 79) for DeAnza. Upper left corner of screen corresponds to 0,0 and lower right corner to 24,79. For mode = O, line and column numbers refer to the x,y coordinates of the overlay channel, lower left corner = 0,0.
Please, note an important detail with the option CURSOR for X11 displays: While additional label strings are entered, keep the cursor (i.e. the mouse) always in the display window, do not move the mouse back into the Midas command window...!!
Also, not all special characters may be supported by the X-Font chosen...

**Examples:** `LABEL/DISPLAY GALAXY 10,0 A 0`
For DeAnza only:
Put the label GALAXY on the image display at line 10, column 0 of the alphanumerics memory in white on black.

```
LABEL/DISPLAY "My star is red" CURSOR OV yellow 1
```

# LABEL/GRAPHIC

For X11 only:
Use the cursor to determine the position where to put the label "My star is red" in the overlay channel in large letters and in yellow color.

```
LABEL/DISPLAY GALAXY CURSOR OV 90
```

For DeAnza only:
Use the cursor to determine the position where to put the label GALAXY in the overlay channel and write it vertically.

```
LABEL/DISPLAY in_a inputi OV ? ?  keyword
```

Write "GALAXY" and "Viva Midas" into the overlay channel at positions 20,20 and 230,100.
Keyword IN_A contains: 'GALAXY Viva Midas ' and keyword inputi holds 20,20,230,100.

---

LABEL/GRAPHIC *core* 15-JUL-1988 RHW

**Purpose:** Write a label on graphics device; the position of the label can be given with the cursor or directly in world coordinates

**Syntax:** `LABEL/GRAPHIC label [x_pos,y_pos[,mm]] [angle] [size] [pos_ind]`

`label` label_string to write up to 40 characters. If the label contains any blanks, it has to be enclosed in double quotes (")

`x_pos,y_pos[,mm]` position of the text in the plot.
If the flag ",mm" is given the label will be drawn x_pos mm. from the left y-axis boundary and y_pos mm. from the bottom x axis boundary.
Else, the position is interpreted as a world coordinate position.
In case NO coordinate pair is entered the graphics cursor will appear, allowing you to enter the position with the left mouse button. Exit with the right mouse button.

`angle` angle for the text direction; defaulted to 0

`size` character multiplication factor; defaulted to 1

`pos_ind` adjustment indicator: 0=text centered on position; 1=text starting at position; 2= text ending at position; default is 0

**Note:** The command allows to overplot text outside the graphics frame. In such a case, if one wants to input coordinates an extrapolation of the x- and/or y-axis should be applied.
The command allows you to re-iterate: as long as the exit button isn't pressed you can choose another position of the text with the left button.

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, ASSIGN/GRAPHICS, OVERPLOT/SYMBOL, OVERPLOT/LINE

**Examples:**

```
LABEL/GRAPH "a:  Sa - Sab" 1.0,0.7 ?  1.5 1

LABEL/GRAPH "b:  Sb - Sbc" 1.0,0.6 ?  1.5 1

LABEL/GRAPH "c:  > Sc" 1.0,0.5 ?  1.5 1

LABEL/GRAPH "All Types" 30.2,20.7,mm ?  1.5 1
```

Write four lines legenda in the plot starting. The first three start on world coordinate 1.0,0.7 to 1.0,0.5; the last label will be put 30.2 mm in x and 20.7 mm. in y from the lower left corner.

LOAD/CURSOR *core* 04-MAR-1985 KB

**Purpose:** Load programmable cursor into the Image Display

**Syntax:** `LOAD/CURSOR curs_table curs_no`

| | |
|---|---|
| `curs_table` | name of table with cursor array, or name,column_label if the standard column :CURSOR should not be used |
| `curs_no` | specifies which cursor should be modified; defaulted to 0 |

**See also:** GET/CURSOR, SET/CURSOR

**Note:** The command LOAD/CURSOR is only supported on DeAnza image processors.
System cursor table files are stored in MID_SYSTAB, your own specific cursor tables should be in MID_WORK.

**Examples:** `LOAD/CURS crossb`
Load contents of column :CURSOR of system table file 'crossb.cur' as cursor array to cursor no. 0 of DeAnza IP8500 system.

# LOAD/IMAGE

## LOAD/IMAGE *core* 27-JUN-1995 KB

**Purpose:** Load image into display device

**Syntax:** `LOAD/IMAGE frame_spec [chanl] [scale] [center] [cuts] [dirs] [fix]`

| | |
|---|---|
| `frame_spec` | frame[,1st_plane..last_plane] with name of frame to be loaded and the first plane and last plane to load, if 'frame' is a 3-dim image |
| `chanl` | memory channel selected for storing the image, defaulted to currently displayed channel |
| `scale` | xscale,yscale[,A] - scaling factors in x,y,average_flag or simply scale[,A] if same scaling factor in x,y;<br>for scaling factors larger than 1, the image is magnified, for s. f. smaller than -1, the image is reduced when loaded.<br>For image reduction (s.f. < -1), lines/pixels are just omitted, unless the average_flag (= A) is added : then the average is taken. Default = 1,1 . |
| `center` | centx,centy - coordinates of image pixels to be put on screen-center, any previously given center pixels (from earlier LOAD/IMA commands) are used or the center of the image will be used as default;<br>the center pixels are entered in the usual MIDAS syntax, e.g. as real world coords. or frame pixels preceded by @.<br>centx or centy may also be set to C,C , if the image should be recentered on the screen |
| `cuts` | optional low, high cutvalues for loading or the method with which to compute these cut values;<br>the method is entered via 'D,method' or 'F,method';<br>D(efault) means that the cuts are only calculated if no valid cuts exist yet (i.e. real descriptor LHCUTS does not exist or LHCUTS(1) .GE. LHCUTS(2)),<br>whereas F(orce),method will always calculate the cuts according to the given method;<br>method is defaulted to 3SIGMA (see Note below);<br>if low, high cuts are given and low cut .GE. high cut, then the cuts are set to min, max of frame (for backward compatibility);<br>thus, setting e.g. the cuts to 0.0,0.0 has the same effect as entering F,MIN (see Note below)<br>if the parameter cutvals is specified, the descriptor LHCUTS is updated accordingly |
| `dirs` | load directives - 2 strings separated by a comma:<br>1. string = U(p) or D(own), for loading the image bottom-up or top-down (e.g. IUE);<br>2. string = N(o) or O(verwrite), for cleaning up image channel before loading or for overwriting existing data without updating the channel data structures;<br>defaulted to Up,No |
| `fix` | fx,fy,sx,sy - pixel numbers of the fixpoint, a point in the image and the screen which should coincide;<br>i.e. the image is positioned on the screen in such a way that frame pixel (fx,fy) is at screen pixel (sx,sy).<br>The pixels are entered as integers, e.g framey in [1,NPIX(2)] and screenx in [0,x-screensize]<br>This overrides the center specifications described above! |

**See also:** VIEW/IMAGE, CUTS/IMAGE, GET/IMAGE, GET/CURSOR, LOAD/LUT, LOAD/ITT SHOW/CH
DISPLAY/CHANNEL, ASSIGN/DISPLAY, COPY/DISPLAY

**Note:** The parameters may also be referenced via:
FRAME= CHANL= SCALE= CENTER= CUTS= DIRS= FIX=
If you are loading the image into the currently displayed image channel, the alphanumeric information is also updated.
If you use the OVERWRITE option (2.string of par. 'dirs' = Over), all subsequent commands will still assume that the original image is loaded; e.g. GET/CURSOR will return incorrect values!
If the keyword MID$SPEC(7:10) is set = CUTS, the command will ask for new cut values in the end and reload the image with these cut values.
If the descr. LHCUTS does not exist and the parameter 'cutvals' is not specified, the min and max of the frame is computed and the cuts are set to mean-3*sigma, mean+3*sigma.
For the calculation of the cut values (if not given explicitly) you may use the following methods:
MIN - use min, max as cuts.
xSIGMA - use mean +- x*sigma as cuts (x = 0,1,...,9).
xSIGMA,ySIGMA - use mean-x*sigma and mean+y*sigma as cuts.
HIGH - use mean-0.1*max and max as cuts. Default is 3SIGMA.
IHAP - use the IHAP algorithm for CCD frames (IHAP batch KDISP) to calculate cuts
For 3-dim frames the planes are counted beginning with 1; the last plane can also be specified via -1. If only the first plane is given, only that one is loaded, and 'frame,all' will load all planes. All relevant planes are loaded into the image channel given as 2nd parameter.
To explore (view) an image in depth (using zoomed subwindows and graphics) use the command VIEW/IMAGE.

**Examples:** `LOAD/IMAGE galaxy 0`
Load image 'galaxy.bdf' into channel 0

`LOAD/IMAGE FRAME=ngc1234 CUTS=0.33,2.34`
Load image 'ngc134.bdf' into currently active channel, use cut values 0.33,2.34

`LOAD/IMAGE FRAME=vaca CENT=@220,@380`
Load image 'vaca.bdf' with x-pixel no. 220 and y-pixel no. 380 of the image placed in the center of the display.
This also shows that you can abbreviate the reference labels...

`LOAD/IMAGE frame=vaca center=47.3,180.3`
Load image 'vaca.bdf' with x-pixel corresponding to world coordinate 47.3 and y-pixel corresponding to world coord. 180.3 of the image placed in the center of the display.

`LOAD/IMAGE frame=toro cuts=D,2sigma`
Load image 'toro.bdf', if no descr. LHCUTS exists or user cuts out of order (i.e. low_cut .GE. high_cut), set the cuts to mean-2*sigma and mean+2*sigma, update descr. LHCUTS.
If instead of the 'D,2sigma' we had 'F,2sigma' then the cuts would always be set and LHCUTS updated accordingly.

`LOAD/IMAGE frame=toro[<,<:@200,@100] fix=1,1,0,0`
Load sub-image of 'toro.bdf' in the lower left corner of the display.

`LOAD/IMAGE ternero,2..3`
Load 2nd and 3rd plane of 3-dim image 'ternero.bdf'. Subsequent cursor readings of the display will refer to frame pixels in the 3rd plane.

`LOAD/IMAGE ternero,4`
Load 4th plane of 3-dim image 'ternero.bdf'.

`LOAD/IMAGE FRAME=becerra SCALE=-3,A`
Load image 'becerra.bdf' into currently active channel, average over a 3*3 box to get the pixel value for loading.

# LOAD/ITT

## LOAD/ITT *core* 24-JUL-1990 KB

**Purpose:** Load an intensity transfer table to Image Display.

**Syntax:** `LOAD/ITT in_specs [chanl] [load_specs]`

| | |
|---|---|
| `in_specs` | table or image,descr<br>(a) table - table name or table,:column if the default column :ITT should not be used<br>(b) image,descr - name of frame and descriptor where the ITT is stored |
| `chanl` | Image Display channel, defaulted to last used channel |
| `load_specs` | section, no. of values - defaulted to 0,256;<br>this parameter is only applicable for DeAnza displays (!) |

**See also:** GET/ITT, MODIFY/ITT, LOAD/LUT, chapter 6 of MIDAS User's Guide

**Note:** The LOAD/ITT command first looks in the current directory for the specified ITT table, if not found it searches for a system ITT table, which are stored in the directory MID_SYSTAB.
ITT columns may only be specified via labels (e.g. :MYLABL) not via numbers (e.g. #3).
For a demonstration of the standard ITTs use "TUTORIAL/ITT" .

**Examples:** `LOAD/ITT jigsaw`
Load contents of column :ITT of system table file jigsaw.itt as ITT into currently active channel

`LOAD/ITT mine,:special 2`
Load contents of column :SPECIAL of my private table file mine.itt as ITT to channel 2

`LOAD/ITT flaco,MIDAS_ITT`
Load contents of descriptor MIDAS_ITT stored in frame flaco.bdf as ITT into currently active channel

LOAD/LUT *core* 10-JAN-1995 KB

**Purpose:** Load colour lookup table into Image Display.

**Syntax:** `LOAD/LUT in_specs [load_specs] [disp_flag] [format]`

| | |
|---|---|
| `in_specs` | table or image,descr<br>(a) table - table name or table,:redlabl,:greenlabl,:bluelabl if the default columns :RED, :GREEN, :BLUE are not used<br>(b) image,descr - name of frame and descriptor where the LUT (as R1,G1,B1,R2,G2,B2,... is stored<br>(c) ASCII file name - each line contains red, green and blue component separated by blanks or comma |
| `load_specs` | section, no. of values - defaulted to 0,256<br>this parameter is only applicable for DeAnza displays (!) |
| `disp_flag` | D(isplay) or N(oDisplay) of LUT bar;<br>if omitted no change in current setup |
| `format` | TABLE or ASCII, for Midas binary table format or ASCII;<br>defaulted to TABLE |

**See also:** GET/LUT, MODIFY/LUT, DISPLAY/LUT, LOAD/ITT, TUTORIAL/LUT
chapter 6 of MIDAS User's Guide

**Note:** The LOAD/LUT command first looks in the current directory for the specified LUT table, if not found it searches for a system LUT table, which are stored in the directory MID_SYSTAB.
LUT columns may only be specified via labels (e.g. :MYLABL) not via numbers (e.g. #3).
If the LUT is an ASCII file, that data values may either be in the intervals [0,255] or normalized to [0.0,1.0].
For a demonstration of the standard LUTs use "TUTORIAL/LUT" .

**Examples:** `LOAD/LUT staircase`
Load contents of columns :RED, :GREEN and :BLUE of system table 'staircase.lut' as LUT.

`LOAD/LUT mine,:special,:green,:myblue`
Load contents of column :SPECIAL, :GREEN and ;MYBLUE of private table 'mine.lut' as LUT.

`LOAD/LUT flaco,MIDAS_LUT`
Load contents of descriptor MIDAS_LUT stored in frame 'flaco.bdf' as LUT.

`LOAD/LUT gordo ? NO`
Load contents of columns :RED, :GREEN and :BLUE of LUT table 'gordo.lut' into the display. Do not display the colour bar at the bottom of the display.

`LOAD/LUT ridiculo.dat ? ? ASCII`
Load contents of ASCII file 'ridiculo.dat' into the display. The LUT components (red, green, blue) per line are either in the interval [0,255] or [0.0,1.0].

# LOAD/OVERLAY

## LOAD/OVERLAY *core* 04-MAR-1985 KB

**Purpose:** Load an overlay table into Image Display

**Syntax:** `LOAD/OVERLAY overlay_table load_specs`

`overlay_table` name of lookup table file for graphics + overlay

`load_specs` section,novals; defaulted to 0,1024

**See also:** SET/OVERLAY, CLEAR/OVERLAY, DRAW/LABEL

**Note:** The command LOAD/OVERLAY is only supported on DeAnza image processors.
System overlay table files are stored in MID_SYSTAB, your own specific overlay tables should be in MID_WORK.

**Examples:** `LOAD/OVER default`
Load overlay table 'default.ovr' into the Display.

## LOAD/TABLE *core* 17-OCT-1983 JDP

**Purpose:** Load table values in the overlay plane of the Image Display

**Syntax:** `LOAD/TABLE table x y [ident] [symbol [size [intens]]]`

| | |
|---|---|
| `table` | table file name |
| `x` | column used as abscissa |
| `y` | column used as ordinate |
| `ident` | optional column used as reference. If included, points with equal identifier value are connected by lines.<br>The table should be sorted by this column. |
| `symbol` | optional integer symbol code:<br>negative - no symbol; 0,2 - square, 1 - circle, 3 - triangle;<br>defaulted to 0 |
| `size` | symbol size in pixels (default is 10) |
| `intens` | intensity for symbol<br>for DeAnza displays the intensity goes from 0 (transparent), 1 (dark gray) to 255 (white);<br>for X11 displays it's really a colour (see Note);<br>defaulted to 255 |

**See also:** PLOT/TABLE, DRAW/RECTANGLE, CLEAR/OVERLAY

**Note:** Coordinate conversion from world or pixel values to screen values is done according to the column units.
The necessary input for the conversion is taken from the currently displayed image, so there must be an image loaded!
The following colours are supported (via name or number) in X11:
Red(3), Green(4), Blue(5), White(2), Black(1), Yellow (6), Magenta(7), Cyan(8) - "colour" 0 is used to erase. All values in [9,255] are interpreted as White.

**Examples:** `LOAD/TABLE coords :X_COORD :Y_COORD`
Draw a small rectangle around the positions where we used GET/CURSOR before (with output to table 'coords.tbl').

# LOCK/KEYWORD

## LOCK/KEYWORD *core* 13-FEB-1992 KB

**Purpose:** Lock specified keyword(s).

**Syntax:** `LOCK/KEYWORD key_list lockno`

| | |
|---|---|
| `key_list` | one or more keywords separated by a comma (no spaces!) |
| `lockno` | a number serving as access code if you want to update any of the locked keywords; a value of 0 (zero) means no lock, i.e. if lockno = 0, you unlock the specified keyword(s) |

**See also:** WRITE/KEYWORD, READ/KEYWORD

**Note:** To modify a locked keyword you have to store the correct lock number into keyword MONITPAR(10) first.

**Examples:** `LOCK/KEY IN_A 4711`
Use locking number 4711 to lock keyword IN_A.

`LOCK/KEY out_a,out_b 0`
Unlock keyword out_a and out_b. This, however, will only work if you had put the current lock number of out_a and out_b into MONITPAR(10) (assuming that both keywords had the same lockno.).

## LOG/OFF *core* 04-JULY-1983 KB

**Purpose:** Disable logging of all terminal I/O in the MIDAS logfile.

**Syntax:** `LOG/OFF`

**See also:** LOG/ON, LOG/TOF, PRINT/LOG, DELETE/LOG

**Note:** The contents of the logfile will not be deleted by this command
Use LOG/ON to resume logging.
By default logging is enabled when starting MIDAS.

**Examples:** `LOG/OFF`
Disable logging in MIDAS from now on.

## LOG/ON *core* 04-JULY-1983 KB

**Purpose:** Enable logging of all terminal I/O in the MIDAS logfile

**Syntax:** `LOG/ON`

**See also:** LOG/OFF, LOG/TOF, PRINT/LOG, DELETE/LOG, PLAYBACK/LOGFILE

**Note:** Logging is enabled by default when starting MIDAS.
The MIDAS logfile is named FORGRxy.LOG (with xy the MIDAS unit you entered when starting MIDAS via INMIDAS (VMS) (or $inmidas (Unix)) and stored in the directory MID_WORK, the MIDAS work directory you specify via SETMIDAS ($setmidas).
The information from the HELP commands is not (!) logged in order to keep the logfile compact. If you want logging also of the on-line help, set keyword LOG(3) to 1. However, the command PRINT/HELP is a simpler way to get a printout of the on-line help.

**Examples:** `LOG/ON`
Enable logging of future MIDAS commands.

## LOG/TOF *core* 04-JULY-1983 KB

**Purpose:** Write a top_of_form (i.e. begin of new page) into the MIDAS logfile.

**Syntax:** `LOG/TOF`

**See also:** LOG/ON, LOG/OFF, PRINT/LOG, DELETE/LOG

**Note:** With this command you can structure your logfile (i.e. make it more readable).

**Examples:** `LOG/TOF`
Enforce next log entry to top of new page.

# MAGNITUDE/CENTER

## MAGNITUDE/CENTER *core* 30-MAY-1995 KB

**Purpose:** Compute the magnitude of the specified object by integrating over the 9 central pixels of it. The local background as integrated over the given background area is subtracted.

**Syntax:** `MAGNITUDE/CENTER [in_specs] [out_specs] [Fsiz,Nsiz,Bsiz] [out_opt]`

`[center_params] [curs_specs] [zw_option]`

`in_specs` input specifications:
(a) CURSOR - if the subimages are chosen via the cursor(s)
(b) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND
(c1) image - if subimage is defined by 'Fszi,Nsiz,Bsiz' around center of the image
(c2) image,xpix,ypix - if subimage is defined by 'Fszi,Nsiz,Bsiz' around pixel (xpix,ypix) of the image
defaulted to CURSOR

`out_specs` output specifications:
(d) table name, may be same as given in in_specs (b), if same table only the magnitude, sigma-mag., sky value and sigma-sky will be stored in columns :MAGNITUDE, :MAG_SIGMA, :SKY and SKY_SIGMA. If new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND and :YEND will be filled; a column labeled :IDENT will automatically be added if not already existing.
(e) descriptor,D if the results should be stored in a real descriptor of the involved image frame; data will be stored as xstart, ystart, xend, yend, magnitude, mag-sigma, sky and sky-sigma
(f) ? , the default, to just display values on the terminal

`Fsiz,Nsiz,Bsiz`
size of flux area, no_man's_land and background (sky) area:
given as world coordinates or pixel numbers, the innermost (flux) square will have a size of 9 pixels so 'Fsiz' is ignored (but must be given!), total area will be a square with size: 9 + 'Nsiz'*2 + 'Bsiz'*2;
defaulted to @9,@2,@2

`out_opt` options for descriptor or table output:
A, append data to a descriptor, else start at the beginning, only applicable to (d)
ID, fill column labeled :IDENT in output table with character string typed in at the terminal, only applicable to (d) with cursor input

`center_params` center_flag,kappa
center_flag = 1 or 0,
if 1 we first find the center of the subimage and shift to it the center of the window we use for the magnitude calculation;
else we use the given subimage to calculate the magnitude,
kappa = factor for kappa-sigma clipping algorithm which is used to calculate the mean background (sky);
defaulted to 0,2. for cursor input, in_specs (a), to 1,2. for (b) + (c1), and to 0,2. for (c2)

`curs_specs` no_curs,drawflg1,drawflg2,max_input:
no_curs = 1 (fixed) for single cursor, a fixed square according to Fsiz,Bsiz,Nsiz given above is used;
drawflg1 = 1/0 for drawing squares in overlay plane or not,
drawflg2 = 1/0 for drawing labels in overlay plane or not,
max_input = max. of cursor inputs.
this parameter is only used if in_specs = CURSOR (a),
defaulted to 1,1,0,9999

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom_window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

**See also:** MAGNITUDE/CIRCLE, MAGNITUDE/RECT, INTEGRATE/APERTURE, CENTER/...

**Note:** The magnitude calculation is done in a two-step procedure:
First, the center of the subimages is computed, which is then used for the magnitude calculations. However, when using the 'image,table' input option (b) and the table contains also the columns :XCEN, :YCEN, then the center of the subimages (as defined by :XSTART, ...) are NOT computed but taken from these columns, instead.
When using the cursor input option (a), 3 squares will be drawn in the overlay plane (if 'drawflg1' of param. 'curs_specs' is set to 1) showing the respective areas for the flux calculation, the no_man's_land and the background/sky area. For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start magnitude calculation on a subimage. To exit set both cursors off and press ENTER . For X-Windows: Press ENTER button on the mouse to start magnitude calculations and draw all the squares. Use the mouse to move cursor window. To exit press the EXIT button on the mouse.
If you also use the 'zw_option', the cursor in the main display is used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
Last results of magnitude calculations are also written into keyword OUTPUTR.
The mean background level is estimated by iterating in a kappa-sigma clipping procedure (max_iter=10) with controllable 'kappa' (via parameter 'center_params').

**Examples:** `MAGNITUDE/CENTER`
Use cursor to define center of subimages and display start- and end-coordinates of subimage as well as magnitude and sky values and associated errors on terminal only

`MAGNITUDE/CENTER CURSOR VALUES,D`
As above but write data to beginning of real descriptor VALUES of displayed frame.

`magnitude/CENTER ? values ? ? 1,3.0`
As 1. example but store data into table 'values.tbl' (will be created by MIDAS), also we recenter the subimage before computing the magnitudes and use a kappa of 3.0.

`MAGNITUDE/CENTER CURSOR values @9,@10,@6 ID`
As above but enter character identifier via keyboard and use different Nsize,Bsize

`MAGNITUDE/CENTER ccd001,sources`
Use columns :XSTART, :YSTART, :XEND, :YEND of table 'sources.tbl' to define the subimages of 'ccd001.bdf', just show results on terminal.

# MAGNITUDE/CENTER

`MAGNITUDE/CENTER ccd001,sources sources`
As above but store results also in same table sources.tbl

`MAGNITUDE/CENTER cursor p7=w`
As 1. example but use zoom window to determine working area via cursor, but we need X11 for that...

`MAGNITUDE/CENTER ccd001,@120,@320`
Use the subimage centered at xpix=120,ypix=320 of 'ccd001.bdf', just show results on terminal.

MAGNITUDE/CIRCLE *core* 30-MAY-1995 KB

**Purpose:** Compute the magnitude of the specified object by integrating over the central area defined by a circular aperture. The local background as integrated over the given background area is subtracted.

**Syntax:** `MAGNITUDE/CIRCLE [in_specs] [out_specs] [Fsiz,Nsiz,Bsiz] [out_opt]`

`[center_params] [curs_specs] [zw_option]`

`in_specs` input specifications, either
(a) CURSOR - if the subimages are chosen via the cursor(s)
(b) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND
(c1) image - if subimage is defined by 'Fszi,Nsiz,Bsiz' around center of the image
(c2) image,xpix,ypix - if subimage is defined by 'Fszi,Nsiz,Bsiz' around pixel (xpix,ypix) of the image
defaulted to CURSOR

`out_specs` output specifications:
(d) table name, may be same as given in in_specs (b), if same table only the magnitude, sigma-mag., sky value and sigma-sky will be stored in columns :MAGNITUDE, :MAG_SIGMA, :SKY and SKY_SIGMA. If new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND and :YEND will be filled a column labeled :IDENT will automatically be added if not already existing.
(e) descriptor,D if the results should be stored in a real descriptor of the involved image frame, data will be stored as xstart, ystart, xend, yend, magnitude, mag-sigma, sky and sky-sigma
(f) ? , the default, to just display values on the terminal

`Fsiz,Nsiz,Bsiz`
size of flux area, no_man's_land and background (sky) area:
given as world coordinates or pixel numbers, the innermost (flux) circle will have a radius of 'Fsiz'/2 pixels, total area will be a circle with diameter: 'Fsiz' + 'Nsiz'*2 + 'Bsiz'*2;
defaulted to @12,@2,@2

`out_opt` options for descriptor or table output:
A, append data to a descriptor, else start at the beginning, only applicable to (d)
ID, fill column labeled :IDENT in output table with character string typed in at the terminal, only applicable to (d) with cursor input

`center_params` center_flag,kappa
center_flag = 1 or 0.
if 1 we first find the center of the subimage and shift to it the center of the window we use for the magnitude calculation;
else we use the given subimage to calculate the magnitude,
kappa = factor for kappa-sigma clipping algorithm which is used to calculate the mean background (sky);
defaulted to 0,2. for cursor input, in_specs (a), to 1,2. for (b) + (c1), and to 0.2. for (c2)

# MAGNITUDE/CIRCLE

`curs_specs` no_curs,drawflg1,drawflg2,max_input
no_curs = 1 or 2, both for a single cursor with 3 circles which have a radius according to Fsiz, Nsiz and Bsiz given above;
but no_curs = 1 inhibits, no_curs = 2 enables the modification of the radius of the circles interactively, as explained below;
drawflg1 = 1/0 for drawing circles in overlay plane or not,
drawflg2 = 1/0 for drawing labels in overlay plane or not,
max_input = max. of cursor inputs.
this parameter is only used if in_specs = CURSOR (a),
defaulted to 1,1,0,9999

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom_window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

**See also:** MAGNITUDE/CENTER, MAGNITUDE/RECT, INTEGRATE/APERTURE, CENTER/GAUSS

**Note:** The magnitude calculation is done in a two-step procedure:
First, the center of the subimages is computed, which is then used for the magnitude calculations. However, when using the 'image,table' input option (b) and the table contains also the columns :XCEN, :YCEN, then the center of the subimages (as defined by :XSTART, ...) are NOT computed but taken from these columns, instead.
When using the cursor input option (a), 3 circles will be drawn showing the respective areas for the flux calculation, the no_man's_land and the background/sky area.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start magnitude calculation on a subimage. Set both cursors on in order to move cursor window. Change cursor size by setting one cursor off, move joystick. To exit set both cursors off and press ENTER. For X-Windows: Press ENTER button on the mouse to start magnitude calculations and draw all the circles. Use the mouse to move the cursor. To exit press the EXIT button on the mouse.
If 'no_curs' = 2, use the arrow keys to change the size of the cursor circles individually. Pressing the function key F1 lets you change the inner circle only, F2 lets you change the middle circle only, F3 lets you change the outer circle only and F4 updates all circles synchronously (default is F4).
If you also use the 'zw_option', the cursor in the main display is used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
Last results of magnitude calculations are also written into keyword OUTPUTR.
It is recommended NOT to change the size of the flux area once you have started in order to integrate always over the same fraction of the stellar profile.
The mean background level is estimated by iterating in a kappa-sigma clipping procedure (max_iter=10) with controllable 'kappa' (via parameter 'center_params').

**Examples:** `MAGNITUDE/CIRCLE`
Use cursor to define center of subimages and display start- and end-coordinates of subimage as well as magnitude and sky values and associated errors on terminal only.
The sizes of the different cursor circles cannot be modified.

`MAGNITUDE/CIRCLE cursor p7=w`
As above but use zoom window to determine working area via cursor, but we need X11 for that...

MAGNITUDE/CIRCLE CURSOR values,D As 1. example but write data to beginning of real descriptor VALUES of displayed frame.

`magnitude/CIRCLE ? values ? ? 1,3.0`
As above but store data into table 'values.tbl' (will be created by MIDAS), also we recenter the subimage before computing the magnitudes and use a kappa of 3.0.

`MAGNITUDE/CIRCLE CURSOR values @20,@10,@6 ID`
As above but enter character identifier via keyboard and use different Fsize,Nsize,Bsize

`MAGNITUDE/CIRCLE ccd001,sources`
Use columns :XSTART, :YSTART, :XEND, :YEND of table 'sources.tbl' to define the subimages of 'ccd001.bdf', just show results on terminal.

`MAGNITUDE/CIRCLE ccd001,sources sources`
As above but store results also in same table 'sources.tbl'

`MAGNITUDE/CIRCLE ccd001,@120,@320`
Use the subimage centered at xpix=120,ypix=320 of 'ccd001.bdf', just show results on terminal.

# MAGNITUDE/RECTANGLE

## MAGNITUDE/RECTANGLE *core* 30-MAY-1995 KB

**Purpose:** Compute the magnitude of the specified object by integrating over the central area defined by a square aperture. The local background as integrated over the given background area is subtracted.

**Syntax:** `MAGNITUDE/RECTANGLE [in_specs] [out_specs] [Fsiz,Nsiz,Bsiz] [out_opt]`

`[center_params] [curs_specs] [zw_option]`

`in_specs` input specifications, either
(a) CURSOR - if the subimages are chosen via the cursor(s)
(b) image,table - if the subimages are defined in a table in the columns labeled :XSTART, :XEND, :YSTART and :YEND,
(c1) image - if subimage is defined by 'Fszi,Nsiz,Bsiz' around center of the image
(c2) image,xpix,ypix - if subimage is defined by 'Fszi,Nsiz,Bsiz' around pixel (xpix,ypix) of the image
defaulted to CURSOR

`out_specs` output specifications:
(d) table name, may be same as given in in_specs (b), if same table only the magnitude, sigma-mag., sky value and sigma-sky will be stored in columns :MAGNITUDE, :MAG_SIGMA, :SKY and SKY_SIGMA. If new table, then in addition to the columns explained above, also the columns labeled :XSTART, :YSTART, :XEND and :YEND will be filled a column labeled :IDENT will automatically be added if not already existing.
(e) descriptor,D if the results should be stored in a real descriptor of the involved image frame, data will be stored as xstart, ystart, xend, yend, magnitude, mag-sigma, sky and sky-sigma
(f) ? , the default, to just display values on the terminal

`Fsiz,Nsiz,Bsiz` size of flux area, no_man's_land and background (sky) area:
given as world coordinates or pixel numbers, the innermost (flux) square will have a size of 'Fsiz' pixels, total area will be a square with size: 'Fsiz' + 'Nsiz'*2 + 'Bsiz'*2 (pixels);
defaulted to @12,@2,@2

`out_opt` options for descriptor or table output:
A, append data to a descriptor, else start at the beginning, only applicable to (d)
ID, fill column labeled :IDENT in output table with character string typed in at the terminal, only applicable to (d) with cursor input

`center_params` center_flag,kappa
center_flag = 1 or 0,
if 1 we first find the center of the subimage and shift to it the center of the window we use for the magnitude calculation;
else we use the given subimage to calculate the magnitude,
kappa = factor for kappa-sigma clipping algorithm which is used to calculate the mean background (sky);
defaulted to 0,2. for cursor input, in_specs (a), to 1,2. for (b) + (c1), and to 0,2. for (c2)

`curs_specs` no_curs,drawflg1,drawflg2,max_input:
no_curs = 1 or 2 for single cursor or cursor rectangle, if single cursor a fixed square according to Fsiz,Bsiz,Nsiz given above is used; for 2 cursors only Nsiz and Bsiz are relevant, Fsize will be determined from the size of the cursor rectangle.
drawflg1 = 1/0 for drawing squares in overlay plane or not,
drawflg2 = 1/0 for drawing labels in overlay plane or not,
max_input = max. of cursor inputs.
this parameter is only used if in_specs = CURSOR (a),
defaulted to 1,1,0,9999

`zw_option` zwindow_flag,zoom;
zwindow_flag = W for zoom_window, N for none,
zoom = initial zoom factor;
only applicable to X11 window displays and option (a),
defaulted to N,4 (see the help of GET/CURSOR for more info about the additional functionality provided in that mode);

**See also:** MAGNITUDE/CENTER, MAGNITUDE/CIRCLE, INTEGRATE/APERTURE, CENTER/GAUSS

**Note:** The magnitude calculation is done in a two-step procedure:
First, the center of the subimages is computed, which is then used for the magnitude calculations. However, when using the 'image,table' input option (b) and the table contains also the columns :XCEN, :YCEN, then the center of the subimages (as defined by :XSTART, ...) are NOT computed but taken from these columns, instead.
When using the cursor input option (a), 3 squares will be drawn in the overlay plane (if 'drawflg1' of param. 'curs_specs' is set to 1) showing the respective areas for the flux calculation, the no_man's_land and the background/sky area.
For DeAnza: Set at least one cursor on, TRACK off. Press ENTER to start magnitude calculation on a subimage. Set both cursors on in order to move cursor window. Change cursor size by setting one cursor off, move joystick. To exit set both cursors off and press ENTER .
For X-Windows: Press ENTER button on the mouse to start magnitude calculations and draw all the squares. Use the mouse to move single cursor or cursor rectangle. Use the arrow keys to change the size of the cursor rectangle. To exit press the EXIT button on the mouse.
If you also use the 'zw_option', the cursor in the main display is only used to define the subwindow to work on. Then move the cursor to the auxiliary (zoom) window and proceed as described above.
When using the out_opt ID for table output, computations proceed after pressing the ENTER button and(!) entering also an identifier of max. 8 characters and hitting RETURN on the keyboard. Otherwise the string IDabcd will be written into column :IDENT with "abcd" the sequence number.
Last results of magnitude calculations are also written into keyword OUTPUTR.
The mean background level is estimated by iterating in a kappa-sigma clipping procedure (max_iter=10) with controllable 'kappa' (via parameter 'center_params').

**Examples:** `MAGNITUDE/RECTANGLE`
Use single cursor or cursor rectangle to define center of subimages and display start- and end-coordinates of subimage as well as magnitude and sky values and associated errors on terminal only.

`MAGNITUDE/RECTANGLE CURSOR VALUES,D`
As above but write data to beginning of real descriptor VALUES of displayed frame.

```
magnitude/RECTANGLE ? values ?  ?  1,3.0
```

# MERGE/TABLE

As above but store data into table 'values.tbl' (will be created by MIDAS), also we recenter the subimage before computing the magnitudes and use a kappa of 3.0.

`MAGNITUDE/RECTANGLE CURSOR values @20,@10,@6 ID`
As above but enter character identifier via keyboard and use different Fsize,Nsize,Bsize

`MAGNITUDE/RECTANGLE ccd001,sources`
Use columns :XSTART, :YSTART, :XEND, :YEND of table 'sources.tbl' to define the subimages of 'ccd001.bdf', just display results on terminal.

`MAGNITUDE/RECTANGLE ccd001,sources sources`
As above but store results also in same table 'sources.tbl' .

`MAGNITUDE/RECT cursor p7=w`
As 1. example but use zoom window to determine working area via cursor, but we need X11 for that...

`MAGNITUDE/RECT ccd001,@120,@320`
Use the subimage centered at xpix=120,ypix=320 of 'ccd001.bdf', just show results on terminal.

---

## MERGE/TABLE *core* 12-OCT-1983 JDP

**Purpose:** Merge table files. The output table will consist of the same columns as the first input table and as many rows as the total set of input tables. The command can be used in connection with the SELECT/TAB command to produce subtables the table(s) specified as input.

**Syntax:** `MERGE/TABLE intable [intable ...] outable`

| | |
|---|---|
| `intable` | input table name(s) |
| `outable` | output table name |

**Note:** none

**Examples:** `MERGE/TABLE TABLE1 TABLE2 TABLE3 OUTPUT`
This will merge 'TABLE1', 'TABLE2', and 'TABLE3' into output table 'OUTPUT'.

MODIFY/AREA *core* 23-SEP-1988 KB

**Purpose:** Remove bad data from a circular pixel-area in an image.

**Syntax:** `MODIFY/AREA [source] [resfram] [degree] [constant] [drawflg]`

| | |
|---|---|
| `source` | defines how the pixel areas we modify will be defined:<br>CURSOR to use displayed frame + cursor rectangle to define pixel areas which are to be modified<br>frame,table to use frame + table to define areas;<br>frame to first load that frame into the display and then use the CURSOR option;<br>defaulted to CURSOR |
| `resfram` | name of result frame, will be a copy of the displayed frame except for the "cleaned" areas;<br>defaulted to input frame or displayed frame |
| `degree` | degree of fitting surface (0,1 or 2); defaulted to 2 |
| `constant` | constant in case of degree = 0 (otherwise not used);<br>defaulted to 0. |
| `drawflg` | 1, the cursor defined rectangle is also put into the overlay channel,<br>2, the modified area is loaded immediately after into the image display,<br>3, 1 + 2 above; defaulted to 3 |

**See also:** MODIFY/PIXEL, MODIFY/ROW, MODIFY/COLUMN, REPLACE/IMAGE

**Note:** 1) For cursor input a cursor rectangle will appear on the image display.
For DeAnza the cursor board should be set up with both cursors on, track off and rate on. Use the joystick to set the cursors around the detail. Press ENTER. To exit, set both cursors off and press ENTER.
For X11 displays use the mouse to move the rectangle and the arrow keys to adjust its size. Press the ENTER button on the mouse. To exit press the EXIT button.
2) For image + table input the table columns labeled :XSTART, :XEND, :YSTART and :YEND are used to define the areas.
All pixels inside the circle defined by the cursor_rectangle or table entry will be replaced by a 2-dim fitted surface.
Currently, the command is NOT working on 1-dim images.

**Examples:** `MODIFY/AREA cursor sunshine`
Use the cursor_rectangle to define the pixel area(s) in the displayed frame and put results into frame sunshine.bdf

`MODIFY/AREA rainbow,cloud sunshine`
Obtain from table cloud.tbl the pixel area(s) of frame rainbow.bdf and put results into frame sunshine.bdf

`MODIFY/AREA rainbow hail`
Load frame 'rainbow.bdf' into the image display. Then, use the cursor rectangle to define the pixel area(s) in the displayed frame and put results into frame 'hail.bdf'.

# MODIFY/COLUMN

## MODIFY/COLUMN *core* 17-OCT-1983 KB

**Purpose:** Modify the pixels of one or two adjacent columns of an image by approximation with a least-squares, second order polynomial using the two columns to the left + right as "good" pixels.

**Syntax:** `MODIFY/COLUMN source_def res_frame [col_type] column_coords`

`source_def` defines how the columns will be defined:
(a) = inframe, then we use inframe and par. 'column_coords' to define columns which are to be modified;
(b) = inframe,intab, then we use inframe and table intab to define columns

`res_frame` output file

`col_type` type (1 char.) of column defect;
C if column-values just have additional constant offset and are otherwise correct, e.g. low-sensitivity columns of CCD
V if column values are corrupted, e.g. saturated columns;
defaulted to C

`column_coords` x-coordinates of columns,
if two adjacent columns are to be worked on, coordinates of both have to be entered (only used when no table involved)

**See also:** MODIFY/ROW, MODIFY/PIXEL, MODIFY/AREA

**Note:** Coords given in command line:
The x-coordinates may either be given as world-coordinates or as pixel no. preceded by the character "@".
Coords given via table:
Columns, which have to be labelled :X, contain x world-coordinates.

**Examples:** `MODIFY/COLUMN gordo flaco V 53.,@100`
Approximate the column with x=53 and column no. 100 of frame gordo.bdf and store result into frame flaco.bdf.
The given columns hold no useful data.

`MODIFY/COLUMN gordo,feo flaco C`
Approximate the columns of frame gordo.bdf with x-coordinates of the columns to be modified stored in the column labelled :X in table feo.tbl. Store result in frame flaco.bdf.
The given columns hold useful data with an unknown constant offset.

MODIFY/CUTS *core* 04-OCT-1995 KB

**Purpose:** Modify cut values of full frame or in cursor selected windows of frame.

**Syntax:** `MODIFY/CUTS [image] [cursor_spec]`

| | |
|---|---|
| `image` | if given, this image will be loaded into the display window first; otherwise the currently displayed image will be used |
| `cursor_spec` | character flag, either N(oCursor) or C(ursor) for modifying the cuts of the full displayed frame or on subframes of the displayed frame which are selected via the cursor rectangle; defaulted to No (i.e. the full displayed frame) |

**See also:** CUTS/IMAGE, LOAD/IMAGE

**Note:** In the 'full frame mode' the histogram of the displayed frame is plotted in the graphics window. With the graphics cursor choose the low and high cut values. These cuts are then used to reload the frame and you can choose again other cuts.
Pressing the EXIT button on the mouse (i.e. the right button) while in graphics cursor mode terminates the command. The descriptor LHCUTS(1,2) of the displayed frame is updated accordingly.
In the 'cursor mode' you select a subframe via the cursor rectangle in the display window. The histogram of the selected subframe is plotted in the graphics window.
With the graphics cursor choose the low and high cut values. These cuts are then used to reload the subframe and you can choose again other cuts.
Pressing the EXIT button on the mouse (i.e. the right button) while in graphics cursor mode lets you move on to select other subframes of the displayed frame.
Pressing the EXIT button on the mouse (i.e. the right button) in the display window when you select the subframes terminates the command.
The last cut values specified are also stored in the real keyword OUTPUTR(1,2) in both modes.

**Examples:** `MODIFY/CUTS`
Modify the cut values of the currently displayed frame and update descr. LHCUTS.

`MODIFY/CUTS ursula cursor`
Load image 'ursula.bdf' and use the cursor rectangle to select subframes. Choose new cut values via the graphics cursor and reload the selected subframes. Save the last specified cut values in keyword OUTPUTR(1,2).

# MODIFY/GCURSOR

## MODIFY/GCURSOR *core* 27-OCT-1989 RHW

**Purpose:** Change data in a line (the input image) interactively via cursor input, by interpolating polynomially between cursor positions. NOTE: The original data will be overwritten.

**Syntax:** `MODIFY/GCURSOR frm_in frm_out y_coord xstart,xend cursors,degree`

| | |
|---|---|
| `frm_in` | input frame name |
| `frm_out` | output frame after modifications; default input frame |
| `y-coord` | y-coordinate of the line to modify (default @1) |
| `xstart,xend` | first and last point on the line to be displayed, entered either as line (pixel) number when prefixed by @, or as real world coordinates (default entire line) |
| `cursors` | no. of cursor positions (<= 100, default 2) |
| `degree` | degree of polynomial fit for the interpolation (default 1) |

**Note:** Use any key except (incl. RETURN), or the mouse enter button to validate a cursor position. When the specified number of cursor positions has been specified the interpolation is done and the data is modified. To display these modified data use the SPACE BAR. To exit use the SPACE BAR or exit button on the mouse once more.

**See also:** PLOT/ROW, PLOT/COLUMN, GET/GCURSOR, MODIFY/PIXELS, MODIFY/ROW, MODIFY/COLUMN, MODIFY/AREA

**Examples:** `MODIFY/GCURSOR GL GLO @1 5,50 2,1`
Modify frame GL, line 1, between the world coordinates 5 and 50, with 2 cursor positions and linear interpolation. The modified date will be stored in frame GLO.

MODIFY/ITT *core* 08-SEP-1989 KB

**Purpose:** Modify the currently active intensity transfer table (ITT).

**Syntax:** `MODIFY/ITT [method] [value] [prflag]`

| | |
|---|---|
| `method` | BAND, for overlaying a one value band on the ITT<br>= ROTATE, for rotating the offset of the ITT<br>= SQEEZE, for squeezing/stretching of the ITT<br>= CONVOLVE, for convolving it with another ITT<br>= CONTRAST, for contrast stretching;<br>defaulted to BAND |
| `value` | value we want to use for method BAND, in [0,255];<br>defaulted to 255 for BAND;<br>= CONT or NO, for method CONVOLVE, for continuous convolution or not;<br>defaulted to NO for CONVOLVE |
| `prflag` | only used for method BAND:<br>Y(es) or N(o) for displaying the intensity interval corrsponding to the chosen band,<br>defaulted to No |

**See also:** LOAD/ITT, GET/ITT, TUTORIAL/ITT, MODIFY/LUT, DISPLAY/LUT

**Note:** Use the joystick (or arrow keys for X11) to modify the ITT(s). To terminate, set cursor #1 to off and press ENTER (or press the exit button of the mouse in X11).
If invalid method or value is entered, the defaults are taken.
Method "CONVOLVE" does not use the cursor, instead the name of the ITT table to convolve the current ITT with is asked for.
For method ROTATE you must start rotating towards the right.
To save your new, modified LUT use the command GET/LUT which will create a copy of the current LUT in MID_WORK.

**Examples:** `MODIFY/ITT BAND 88`
Put a band with pixel value 88 over current ITT.

# MODIFY/LUT

## MODIFY/LUT *core* 07-SEP-1989 KB

**Purpose:** Modify the currently active colour lookup table (LUT).

**Syntax:** `MODIFY/LUT [method] [colour] [prflag]`

| | |
|---|---|
| `method` | BAND, for overlaying a one colour band on the LUT<br>= ROTATE, for rotating the offset of one or all colours of LUT<br>= GRAPH, for changing interactively slope and intercept of LUT<br>= SQEEZE, for squeezing/stretching of LUT and moving the offset<br>= HSI, for modifying in H(ue), S(aturation),I(ntensity) space;<br>defaulted to BAND |
| `colour` | colour(s) we want to work with:<br>for BAND: RED, GREEN, BLUE, WHITE, DARK, YELLOW, PINK, BROWN, ORANGE and VIOLET (default = RED)<br>for ROTATE: RED, GREEN, BLUE or ALL (default = ALL)<br>for GRAPH: RED, GREEN, BLUE or ALL (default = ALL)<br>for SQUEEZE not applicable - all colours are affected<br>for HSI: HUE, SATURATION or INTENSITY (default = HUE) |
| `prflag` | only used for method BAND:<br>Y(es) or N(o) for displaying the intensity interval corrsponding to the chosen band, defaulted to No |

**See also:** MODIFY/ITT, TUTORIAL/LUT, DISPLAY/LUT, GET/LUT, LOAD/LUT

**Note:** Use the joystick (or arrow keys if X11) to modify the LUT. To terminate, set cursor #1 to off (or press exit button on the mouse if X11).
If invalid method or colour is entered, the defaults are taken.
For method ROTATE you must start rotating towards the right.
To save your new, modified LUT use the command GET/LUT which will create a copy of the current LUT in MID_WORK.

**Examples:** `MODIFY/LUT ROTATE GREEN`
Rotate only the green colour of current LUT.

MODIFY/PIXELS *core* 23-SEP-1988 KB/RNH

**Purpose:** Modify pixel values on the currently displayed image by approximation over the surrounding area

**Syntax:** `MODIFY/PIXELS [source] [resfram] [arfacts] [xdeg,ydeg,niter]`
`[drawflg] [noise]`

| | |
|---|---|
| `source` | defines how the pixel areas we modify will be defined:<br>CURSOR to use displayed frame + cursor rectangle to define pixel areas which are to be modified<br>frame,table to use frame + table to define areas;<br>frame to first load that frame into the display and then use the CURSOR option;<br>defaulted to CURSOR |
| `resfram` | name of result frame, will be a copy of the displayed frame except for the "cleaned" areas;<br>defaulted to input frame or displayed frame |
| `arfacts` | factors to determine size of surrounding area from cursor area; e.g. if cursors define a 5x5 pixel area, the surface used in the interpolation will be 15*15;<br>max. size of surrounding area is 60x60 pixels;<br>defaulted to 3,3 |
| `xdeg,ydeg,niter` | degree of fitting polynomial in x, y and no. of iterations;<br>defaulted to 2,2,5 |
| `drawflg` | 1, the cursor defined rectangle is also drawn into the overlay channel;<br>2, the modified area is loaded immediately after into the image display;<br>3, 1 + 2 above; defaulted to 3 |
| `noise` | Y or N, whether the replaced region is to have artificial, Gaussian noise added to make it look more natural;<br>defaulted to Y |

**See also:** MODIFY/AREA, MODIFY/ROW, MODIFY/COLUMN, REPLACE/IMAGE

**Note:** 1) Cursor input:
For a DeAnza image display, both cursors have to be on, TRACK off and RATE on. A cursor rectangle will appear on the DeAnza monitor. Use the joystick to move the cursors. Switching one cursor on and off enables you to modify the size of the cursor rectangle. Press ENTER to work on indicated rectangle.
To exit, set both cursors off + press ENTER.
For X11 windows, use the mouse to move the cursor rectangle and the arrow keys on the keyboard to adjust the size. The leftmost button on the mouse is the ENTER button. The next button to the right of the ENTER button is the EXIT button (the RETURN key also serves as EXIT button).
All pixels inside the cursor_rectangle will be interpolated. Interpolation is done with a 2-dim surface fitted to pixels around the cursor_rectangle. A kappa*sigma clipping is applied if "iter" > 0 to delete unwanted point sources from the surface.
The size of the cursor rectangle will be the one from the last usage of the cursor rectangle and could be too large (cf. the description of par. 'arfacts' above). So you may have to modify that size first.
2) Table input:
Columns which have to be labeled XSTART, XEND, YSTART and YEND are used to define the areas.
Currently, the command is NOT working on 1-dim images.

# MODIFY/ROW

**Examples:** `MODIFY/PIX ? zacatecas`
Use the cursor rectangle to define the pixel area(s) in the displayed frame and put results into frame 'zacatecas.bdf'.

`MODIFY/PIX delicias,torreon chihuahua`
Obtain from table 'torreon.tbl' the pixel area(s) of frame 'delicias.bdf', put results into 'chihuahua.bdf'.

`MODIFY/PIXELS veracruz monterrey`
Load frame 'veracruz.bdf' into the image display. Then, use the cursor rectangle to define the pixel area(s) in the displayed frame and put results into frame 'monterrey.bdf'.

---

MODIFY/ROW *core* 17-OCT-1983 KB

**Purpose:** Modify the pixels of one or two adjacent rows of an image by approximation with a least-squares, second order polynomial using the two rows above + below as "good" pixels.

**Syntax:** `MODIFY/ROW source_def res_frame row_type row_coords`

| | |
|---|---|
| `source_def` | defines how the rows we want to modify will be defined:<br>= inframe - use inframe + row_coords to define the rows which are to be modified;<br>= inframe,table - use inframe + table to define rows |
| `res_frame` | output file |
| `row_type` | type of row replacement;<br>= C if row-values just have additional constant offset and otherwise contain correct data;<br>= V if row values are corrupted, e.g. saturated rows;<br>defaulted to C |
| `row_coords` | y-coordinates of rows, if two adjacent rows are to be worked on, the coords of both rows have to be entered;<br>(only used when no table involved) |

**See also:** MODIFY/COLUMN, MODIFY/PIXEL, MODIFY/AREA, REPLACE/IMAGE

**Note:** Coords given in command line:
The y-coordinates may either be given as world-coordinates or as pixel no. preceded by the char. "@".
Coords given via table:
Columns which have to be labelled :Y, contain y world-coordinates.

**Examples:** `MODIFY/ROW hector achilles V 53.,@100`
Approximate the row with y=53.0 and row no. 100 of frame 'hector.bdf' and store results into frame 'achilles.bdf'. The given rows hold no useful data.

`MODIFY/ROW aguila,nopal vibora C`
Approximate the rows of frame 'aguila.bdf' with y-coordinates of the rows to be modified stored in the column labeled :Y in table 'nopal.tbl'. Store result in frame 'vibora.bdf'. The given rows hold useful data with an unknown constant offset.

NAME/COLUMN *core* 12-OCT-1983 JDP + FO

**Purpose:** Redefines the label, unit or format of table columns.

**Syntax:** `NAME/COLUMN table column [new-column] [unit] [format]`

| | |
|---|---|
| `table` | the table name |
| `column` | the column reference |
| `new-column` | the optional new label for the column |
| `unit` | optional units included in double quotation marks. Blanks are used by default |
| `format` | format associated to the column according to the FORTRAN-77 rules with some extensions. The format is used by default when the table is displayed (commands READ/TABLE and PRINT/TABLE). Possible formats are :<br>for characters: Aww<br>for integers: Iww iww<br>Xww Oww for hexa / octal<br>Tww.dd tww.dd for Date+Time<br>(seconds since 1970)<br><br>for floating: Fww.dd fww.dd<br>Eww.dd eww.dd<br>Gww.dd<br>Rww.dd rww.dd for Right Ascensions<br>Sww.dd sww.dd for Sexagesimal (decl.)<br>Tww.dd tww.dd for Date+Time (JD)<br>Zww.dd zww.dd zero-filled |

**See also:** CREATE/COLUMN

**Note:** It is not possible to CHANGE the TYPE of a column. However the command COPY/TT will allow you to transform a column of a certain type into a column of an another type.

**Examples:** `NAME/COLUMN mytable #2 :RADVEL "km.s-1" E12.3`
This command will give to the column 2 of the table mytable the label RADVEL

`NAME/COLUMN mytable :RADVEL :VELOCITY`
This command will give the label VELOCITY to the column of the table mytable which has the label RADVEL

`NAME/COLUMN mytable #2 "km.s-1" E12.3`
This command will give to the column 2 of the table mytable the units "KM.SEC-1". The format E12.3 will be used to display the values of that column

# NORMALIZE/SPECTRUM

## NORMALIZE/SPECTRUM *core* 01-NOV-1986 DB

**Purpose:** Approximate continuum of 1-D spectra for later normalization.

**Subject:** Spectroscopy, flux calibration, spectral analysis.

**Syntax:** `NORMALIZE/SPECTRUM inframe outframe [mode] [table] [batch_flag]`

| | |
|---|---|
| `inframe` | name of input frame |
| `outframe` | name of frame to hold the fit |
| `mode` | GCURSOR, to start from scratch, input from graphics cursor<br>= ADD, to add new points to existing fit, input with cursor<br>= DELETE, to delete points from existing point (with cursor)<br>= TABLE, to take positions and bin widths from "table" and integrate in "inframe" over corresponding bins. Default: GCURSOR |
| `table` | intermediate working table with cursor positions defining wavelengths and bin widths. In mode TABLE the flux in "inframe" is integrated over bins provided with "table". "table" must contain one column labeled :X_AXIS (with the wavelengths) and another one labeled :BIN_WIDTH (with the bin widths). Default: TABLE |
| `batch_flag` | if equal to Y all plots will be suppressed. Default: N |

**Output:** Intermediate table FIT1D is used, with columns :X_AXIS and :Y_AXIS.

**Note:** All data is internally written to table FIT1D on which the actual spline is to be made. Mode GCURSOR is self-evident. Enter data with cursor and ENTER key. Exit: space bar. The wavelengths are taken as the center of the bins. For the integration, a rectangular "transmission curve" is assumed. The points determined are plotted as is the fit derived from them. Modes ADD and DELETE can be used to interactively edit table FIT1D until a satisfactory fit is achieved. To delete a point, reply "*" to the question asked on your session terminal. NOTE that only the latest fit made with this command can be edited!

Note that the actual normalization needs, then, to be done like COMPUTE/IMAGE normalized = inframe/outframe.

**Examples:**

```
NORMALIZE/SPECTRUM RAW FIT
```

start a fit of image RAW, input is expected from graphics terminal, result to be written to new image FIT

```
NORMALIZE/SPECTRUM RAW FIT D
```

delete some ill fitting points (use graphics cursor), a new fit (new image FIT) will be made

```
NORMALIZE/SPECTRUM SPECTRUM CONTINUUM T LAMBIN
```

use wavelengths and bin width in table LAMBIN to accordingly integrate the flux in image SPECTRUM

```
COMPUTE/IMAGE
```

## OPEN/FILE *core* 23-SEP-1992 KB

**Purpose:** Open an ASCII file for reading or writing.

**Syntax:** `OPEN/FILE filename flag file_control_key`

`filename` name of ASCII file (including file type, if any)

`flag` READ, WRITE or APPEND, if opening the file for reading, writing (i.e. creating a new file) or appending;
defaulted to READ

`file_control_key`
name of existing integer keyword of at least 2 elements to receive in its 1st element the file-id of the opened file; subsequent READ/FILE and WRITE/FILE commands will store the no. of chars. read or written into the 2nd element of that keyword;
the file-id is then used in the READ, WRITE and CLOSE/FILE commands

**See also:** CLOSE/FILE, READ/FILE, WRITE/FILE

**Note:** The file-id is a number greater than 0. If the opening of the file failed a value of -1 is stored in the keyword. No error message is displayed, so it is the responsibility of the user to check that value. Opening a file for writing means creating a new file.

**Examples:** `OPEN/FILE apumanque.dat write fctrl`
Create an ASCII file 'apumanque.dat' and store the file-id in the first element of the integer keyword 'fctrl' (which must have been created before with at least two elements).

`OPEN/FILE macul.ascii ?  OUTPUTI`
Open ASCII file 'macul.ascii' for reading and store the file-id in integer keyword 'outputi' (as the first element).

`OPEN/FILE alfonso.asc A inputi`
Open existing ASCII file 'alfonso.asc' for appending records in the end of the file and store the file-id in keyword inputi(1).

# OUTTAPE/FITS

## OUTTAPE/FITS *core* 14-JAN-1994 PJG

**Purpose:** Write frames from MID_WORK to magtape or disk in FITS format.

**Syntax:** `OUTTAPE/FITS [catalog[,list]] device [flags] [density,block] [type]`

`catalog,list` name of catalog containing the names of the frames that will be written to magtape (disk). A list of file numbers in the catalog can be given. Default is all files in all catalog which are set with the SET/xCAT command. The default for list is file 1-9999.
It is also possible to specify a single file name but then its extension must be given.

`device` logical tape unit (e.g. TAPE1), physical tape unit (e.g. /dev/nrst8 or host:/dev/nrst1) or prefix of file name on disk. The default extension for disk files is '.mt'. It is possible to specify a full file name with extension if only one is written.

`flags` 3-character flag: append, display and cut. append_flag: A(ppend) file to tape, R(ewind) i.e. write from start, N(o append) i.e. write from current location.
display flag: F(ull), S(hort) or N(one)
cut flag: C(ut) using LHCUTS(3-4) for frames or display format for tables.
Defaulted to NSN (NOTE: that is NO APPEND mode).

`density,block` density_of_magtape,blocking_factor, defaulted to 6250,10. The tape density is used only for 1/2 inch tapes.

`type` 1-character flag: FITS format
type flag : B(asic) FITS e.g. BITPIX=8,16 or 32 for frames and ASCII for tables. O(riginal) data format i.e. the FITS format closest to frame format including BITPIX=-32,-64 (IEEE-FP) and BINTABLE for tables.
defaulted to O

**See also:** CREATE/xCAT, SET/xCAT, INTAPE/FITS

**Note:** The default is NO rewind mode which will write new files from the current position of tape. This may erase previous information on the tape - be careful and check!
The different parameters may be referenced in any order via CATALOG= DEVICE= FLAGS= DENS,BLCK= TYPE=
If writing to magtape, a tape (with write-ring) must be loaded on a magnetic tape unit before executing the command.
To access remote tape drives, the MIDAS tapeserver demon must be installed and run on the host in question.
MIDAS frames are normally stored as floating point numbers. Such files by default be written using the BITPIX=-32 i.e. IEEE floating point format. Please, check if the FITS reader you want to read the data with has been upgraded to include this. If that is not the case, use the TYPE=B to force OUTTAPE to write a tape conforming to Basic FITS.
FITS files are written with a physical blocking factor of 10 by default. Some old FITS readers (e.g. IHAP) may NOT accept that and require an explicit blocking factor of 1.

**Examples:** `OUTTAPE/FITS * TAPE2 R`
write all files in all catalogs which have been SET in FITS format (in original format e.g. 32 bit IEEE floating point) to the magnetic tape mounted on unit 'TAPE2' (from beginning of tape on) and with short display. If both image, ascii, table and fit catalogs are set all the files they contain will be written.

```
OUTTAPE/FITS mycat DEVICE=tapedat
```

write all images with an entry in the 'mycat' catalog (file mycat.cat) in FITS format (in original format e.g. 32 bit IEEE floating point) to the DAT cartridge tape mounted on unit 'tapedat' (from current position on tape) and with short display

`OUTTAPE/FITS caspeca /dev/nrst8 AN`
skip to end of local magtape with physical name /dev/nrst8, indicated by two consecutive EOFs, append all files from catalog caspeca (file caspeca.cat) in FITS format, do not display FITS header

`OUTTAPE/FITS caspeca,101-220 TAPE8MM AN`
as above but process only the files with entry numbers in the interval [101,220] from 'caspeca' image catalog writing them to the device 'TAPE8MM'.

`OUTTAPE/FITS caspeca,101-220 obs:/dev/nrst8 A TYPE=B`
as above, but writes FITS files conforming to the Basic FITS format (i.e. BITPIX=8,16 or 32) to tape mounted on the tape unit /dev/nrst8 on host 'obs'. NOTE: MIDAS tape setver must run on this host.

`OUTTAPE/FITS caspeca,101-220 data AN`
as above, but convert files to FITS format files and store them in files data0101.mt to data220.mt

`OUTTAPE/FITS galaxy.bdf galaxy.fits`
write the frame galaxy.bdf out in FITS format to the disk file galaxy.fits.

# OVERPLOT/AXES

## OVERPLOT/AXES *core* 21-SEP-1990 RHW

**Purpose:** Overplot a coordinate box with tickmarks and labels

**Syntax:** `OVERPLOT/AXES [x_axis_spec] [y_axis_spec] [x_sc,y_sc[,x_off,y_off]]`

`[x_lab] [y_lab]`

`OVERPLOT/AXES [coord_str] [x_lab] [y_lab]`

`x_axis_spec` x-axis specification: start value, end value, distance between the big tickmarks, distance between the small tickmarks. Default: use the manual setting (by the SET/GRAPHICS command); if not present, use a linear axis from 0.0 to 1.0. The start value can be smaller than the end value.If distance between small tickmarks is less than 0, a logarithmic axis will be plotted, running from 10**start_value to 10**end_value. At least the start and end values should be given. If large tickmark distance fails a (hopefully) sensible default will be used. If small tickmark distance fails no small tickmarks will be plotted.

`y_axis_spec` specfication for the y-axis; see above.

`coord_str` area in the displayed frame in the standard MIDAS notation. This option only works on workstations, if the display window has been assigned as the plot device and a frame has been loaded! The default is a frame box around the whole frame.

`x_sc,y_sc,x_off,y_off`

scale or size in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`x_lab` label for the x-axis; default is no label

`y_lab` label for the y-axis; default is no label

**Note:** The command can draw axes on all graphic output devices, including the display window in workstations. Depending on the assigned graphic device (with ASSIGN/GRAPHICS) the command decides in which mode it will run.
In case the output graphics device is the graphics window, terminal or a hardcopy device the command will always run in the first mode.
If the assigned output graphics device is the display window AND if the first input parameter is given as a MIDAS coordinate string (i.e. starting with '[') the command will run in the second mode (see below) and will be drawn around the LOADED frame. However, with the display window assigned, if the first parameter is not a MIDAS coordinate string the command will run in the first mode. In the second mode, if (part of) the frame comes close to display window boundary part of the coordinate frame will fall outside the window. In that case one can either center the frame, or give a somewhat smaller area to draw the coordiates around.
After finishing, the coordinates of the axes are stored in the MIDAS keyword area, all previous axes settings are overwritten, and the user can use the 'manual' plot mode. This command gives the same result as first setting the x- and y-axis manually (using SET/GRAPHICS), followed by a PLOT/AXES command without specifying the x and y axes.

**See also:** ASSIGN/GRAPHICS, SET/GRAPHICS, SHOW/GRAPHICS, PLOT/AXES

**Examples:** `OVER/AXES 0,100,25,5 1,3,0,-1 -100,-100,50,25 "V_Hel" "Flux"`
Overplot a box with a linear scale in the x and a logarithmic one in the y direction; the size of the frame will be 100 times 100 mm. The box will be put 50 mm from the left border of the graphics and 25 mm from the bottom.

`LOAD/IMAGE spiral`
Load a frame

`ASSIGN/GRAPHICS display`
Assign the display window to the graphics device

`OVER/AXES [@10,@10:@210,@305] "Right Ascension" "Declination"`
Plot a coordinate frame around the displayed frame.

---

## OVERPLOT/COLUMN *core* 04-SEP-1991 RHW

**Purpose:** Overplot a column of a frame on a graphic device

**Subject:** Graphics, frames

**Syntax:** `OVERPLOT/COLUMN frame [x_coord] [y_sta,y_end] [offset] [l_type]`

| | |
|---|---|
| `frame` | image file name |
| `x_coord` | column number or world coordinate in the frame; default the first column. |
| `y_sta,y_end` | can be: (a) first and last pixel coordinate; (b) start, end in world coordinates on the column. Default is either the manual setting done by the command SET/PLOT (if present), or the whole column. |
| `offset` | offset in pixel intensity units; default 0.0 |
| `l_type` | line type to be used; default is the currently enabled line type (use SHOW/PLOT). |

**Note:** none

**See also:** SET/PLOT, SHOW/PLOT, PLOT/COLUMN, PLOT/ROW, OVERPLOT/ROW

**Examples:** `OVERPLOT/COLUMN myframe @256 @10,@100 100`
This command overplots from image myframe.bdf column 256, pixels 10 to 100 with an offset of 100 units.

# OVERPLOT/COLUMN

## OVERPLOT/COLUMN *core* 04-SEP-1991 RHW

**Purpose:** Overplot a column of a frame on a graphic device

**Subject:** Graphics, frames

**Syntax:** `OVERPLOT/COLUMN frame [x_coord] [y_sta,y_end] [offset] [l_type]`

| | |
|---|---|
| `frame` | image file name |
| `x_coord` | column number or world coordinate in the frame; default the first column. |
| `y_sta,y_end` | can be: (a) first and last pixel coordinate; (b) start, end in world coordinates on the column. Default is either the manual setting done by the command SET/GRAPHICS (if present), or the whole column. |
| `offset` | offset in pixel intensity units; default 0.0 |
| `l_type` | line type to be used; default is the currently enabled line type (use SHOW/GRAPHICS). |

**Note:** none

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, PLOT/COLUMN, PLOT/ROW, OVERPLOT/ROW

**Examples:** `OVERPLOT/COLUMN myframe @256 @10,@100 100`
This command overplots from image myframe.bdf column 256, pixels 10 to 100 with an offset of 100 units.

## OVERPLOT/CONTOUR *core* 09-JUN-1987 RHW

**Purpose:** Overplot contour map of 2-dim. frame with a smoothing option

**Subject:** Graphics, frames, contour maps

**Syntax:** `OVERPLOT/CONTOUR frame [coord_str] [contours] [c_type] [sm_par]`

| | |
|---|---|
| `frame` | name of the frame |
| `coord_str` | area to be overplotted in standard MIDAS notation (see MIDAS Users Guide, Volume A, Chapter 3). Default is the manual setting done with the SET/GRAPHICS command (if present), or the whole area. |
| `contours` | contour values; input can be given as cstart:cend:cincr or cnt1,cnt2,cnt3..., or any combination of these two possibilities separated by a comma. (Default 1.0) |
| `c_type` | NEG, ODD, or LTYPE: determines the line type(s) used. NEG will draw negative contours dashed and positive ones solid; ODD will draw odd contours dashed and even ones solid; If LTYPE is specified contours will be drawn with the line type specified with SET/GRAPHICS. Default is NEG |
| `sm_par` | smoothing parameter i.e. box width in pixels; default is 1 |

**Note:** The coordinates for the area are optional. If one specifies a "?", and the manual frame setting(s) for the x- and y-axis are used these settings will be used; else the whole area will be taken. If a "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already.
For the DEANZA to pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.
For XWindows use the mouse to move the cursor, the leftmost mouse button or the RETURN key is the ENTER button, the second left mouse button serves as EXIT button. The cursor cross may not be visible at first. Move the mouse to the lower left corner of the graphics window to grab the cursor.
The size of the plot area (area within the rectangular cursor) can be be changed by fixing one of the two cursor corners (DEANZA) or by using the arrow keys on the keyboard (XWindows).
The maximum area that can be plotted in 512 by 512 pixels.

**See also:** PLOT/CONTOUR, SET/GRAPHICS, ASSIGN/GRAPHICS

**Examples:** `PLOT/CONTOUR spiral [@281,@281:@320,@320] ? 1:5:0.25 LTYPE 3`
Make a contour plot of the frame "spiral". Use default scales and draw the contours 1, 1.25, 1.5, ...till 5. Use the line set by the command SET/GRAPH. Use a smoothing box of 3x3 pixels.

`OVERPLOT/CONTOUR spiral [@281,@281:@320,@320] 6,7,8 ODD 3`
Do an overplot of the same frame, however, now for the contours 6, 7, and 8. Contours 6 and 8 will be dashed, contour 7 solid.

# OVERPLOT/DESCRIPTOR

## OVERPLOT/DESCRIPTOR *core* 9-JUN-1987 RHW

**Purpose:** overplot the contents of a descriptor

**Subject:** Graphics, frames, descriptors

**Syntax:** `OVERPLOT/DESCRIPTOR frame [descr] [start,end] [offset]`

| | |
|---|---|
| `frame` | name of data frame |
| `descr` | name of the descriptor; default is HISTOGRAM |
| `start,end` | first and last data point to be overplotted; end can be smaller than start; defaults are the first and the last valid data points in the descriptor. |
| `offset` | offset value in y direction (world coordinate units); default 0.0 |

**Note:** none

**See also:** PLOT/DESCRIPTOR, READ/DESCRIPTOR, WRITE/DESCRIPTOR, COPY/DD COPY/DK, COPY/KD, SET/GRAPHICS, ASSIGN/GRAPHICS

**Examples:** `OVERPLOT/DESCRIPTOR YOURFRAME histogram`
This will overplot the descriptor histogram associated with the frame 'YOURFRAME'. The descriptor must exist. In this case, it should have been created previously by the command STATISTICS/IMAGE YOURFRAME.

`OVERPLOT/DESCRIPTOR YOURFRAME histogram ? 10`
Similar as previous example only now the elements 100,150 will be plotted with an offset of 10 units.

## OVERPLOT/ERROR *core* 3-DEC-1986 RHW

**Purpose:** Overplot a table error column

**Subject:** Graphics, tables, errors

**Syntax:** `OVERPLOT/ERROR table [col1] [col2] col3 [direct] [bar]`

| | |
|---|---|
| `table` | name of table file |
| `col1` | column in abscissa (x-axis); default is sequence number, however see the note below. |
| `col2` | column in ordinate (y-axis); default is sequence number, however see the note below. |
| `col3` | error column for either the x or y values |
| `direct` | direction of the error bar to be plotted (default 6). The error bars are drawn in the direction (K-1) * 90 deg., where K is the input parameter for "direct". In addition, direct = 5 will plot error bars at both sides of the data points in the x-directon; direct = 6 does the same in the y-direction and which the default. |
| `bar` | option to draw or to avoid the small cross bar at the end of the error bar. Default is 'Y': small ending bar will be drawn. |

**Note:** Only one of the column input parameters can be defaulted to sequence number. So, 'table #1 ?' and 'table ? #2' are both valid input parameter strings, 'table ? ?' is not.
When the user gives a '?' for one of the first two columns the values the corresponding coordinate will be assumed to be the sequence number of the row. At least one column should be given, either as the first or the second parameter.
The functions LOG and LN, decimal and natural logs respectively, can be applied to the columns to be plotted.

**See also:** PLOT/TABLE, OVERPLOT/TABLE, SET/GRAPHICS, ASSIGN/GRAPHICS

**Examples:**

```
PLOT/TABLE MYTABLE :MAG LOG(:TEMP) :TEMPER1 1
```
plot two columns of table MYTABLE

```
OVERPLOT/ERROR MYTABLE :MAG LOG(:TEMP) :TEMPER2 3
```
overplot error bar left from data point

```
OVERPLOT/ERROR GALAXY :MHI_L ? :ERROR 6
```
gives the same result as

```
OVERPLOT/ERROR GALAXY :MHI_L ? :ERROR
```

# OVERPLOT/GRAY

## OVERPLOT/GRAY *core* 06-APR-1988 RHW

**Purpose:** Overplot gray scale map of 2-dim. frame (with a smoothing option)

**Subject:** Graphics, frames, gray scale maps

**Syntax:** `OVERPLOT/GRAY frame [coord_str] gray_lev [sm_par] [gray_ness]`

`[options]`

| | |
|---|---|
| `frame` | name of the frame |
| `coord_str` | area to be plotted in standard MIDAS notation (see MIDAS Users Guide, Volume A, Chapter 3). Default is the manual setting done with the SET/GRAPHICS command (if present), or the whole area. |
| `gray_lev` | gray levels; in case of a continuous gray scale plot two values are required (cnt1,cnt2); in case of stepwise increment of the gray scale plot input can be given as<br>cstart:cend:cincr or cnt1,cnt2,cnt3 or any combination of these two possibilities separated by a comma; default is 0.0,1.0 for a continous gray scale |
| `sm_par` | smoothing parameter i.e. box width in pixels; default is 1 |
| `gray_ness` | grayness parameter between 0.0 and 1.0; the value 1 gives the maximum blackness; 0 means white; default is 1.0 |
| `gray_opt` | plot options to be specified as option1,option2,...; possible options are: LOG (logarythmicly increasing grayness) or LIN (linear increasing grayness), STEP (stepwise gray scales) or CONT (continously increasing gray scales), NEG (change sign of data points) or POS (leave signs unchanged) and ABS (take absolute values) |

**Note:** The coordinates for the area are optional. If one specifies a "?", the area taken will be for the manual frame settings for the x- and y-axis. If a "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already. To pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.

**See also:** PLOT/GRAY, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES, PLOT/CONTOUR, OVERPLOT/CONTOUR, PLOT/VECTOR, OVERPLOT/VECTOR

**Examples:**
```
PLOT/GRAY spiral [@281,@281:@320,@320] ?  1,5
OVERPLOT/GRAY spiral [@281,@281:@320,@320] 1:5 3 1. ABS,LOG
OVERPLOT/GRAY spiral [@281,@281:@320,@320] ?  1:5:0.5 ?  .75 STEP
```

## OVERPLOT/GRID

*core* 17-SEP-1988 RHW

**Purpose:** Overplot a grid on an existing coordinate box

**Subject:** Graphics, plot grid, tickmarks

**Syntax:** `OVERPLOT/GRID [grid_type] [xy]`

| | |
|---|---|
| `grid_type` | grid_type; can be "LARGE" or "SMALL". LARGE means that the grid will be put over the plot by connecting the large tickmarks. SMALL means that the small and large tickmarks will be connected. Default is LARGE |
| `xy` | connect the tickmarks on the x-axis (input X), on the y-axis (input Y) or both. Default is both tickmarks (=XY). |

**Note:** none

**See also:** PLOT/AXES, OVERPLOT/AXES, SET/GRAPHICS, ASSIGN/GRAPHICS

**Examples:** `PLOT/COLUMN myframe @125 @100,@440`
Plot the data of a column in the frame myframe

`OVERPLOT/GRID L XY`
Overplot a grid connecting the large ticks of the coordinate box both on the x- and y-axis.

# OVERPLOT/HISTOGRAM

## OVERPLOT/HISTOGRAM *core* 10-JUN-1987 RHW

**Purpose:** Overplot a histogram of a column in a table or Overplot the histogram of given image

**Syntax:** `OVERPLOT/HISTOGRAM tab col [offset] [log] [opt] [bin[,min[,max]]]`

`[exc] [log] [opt]`

`or`

`OVERPLOT/HISTOGRAM frame [offset] [log] [opt]`

| | |
|---|---|
| `tab` | name of table file |
| `col` | column reference |
| `offset` | offset in the ordinate direction (default 0.0) |
| `bin,min,max` | bin size, minimum, and maximum values to be included in the histogram. Default bin size depends on the actual dynamic range of the data values; default minimum is the minimum of the column values; similar for maximum. |
| `exc` | plot low and high excess; default YY |
| `log` | flag for logarithmic scale; can be LIN (default) LOG or LN. |
| `opt` | histogram type, hashing spacing and hashing angle; default: standard staircase plot, no hashing done.<br>The histogram type is determined by an integer number: 0: simple staircase;<br>1: staircase steps joining the x axis;<br>=> 2: data points will joint the x axis with boxes which are determined by the number (2 = zero width)<br><br>In case the hashing spacing is given a small number (e.g. 0.01) the histogram will be filled completely. A reasonable number is 1. |
| `frame` | name of image frame |

**Note:** In case the second parameter is defaulted, the command assumes a frame as input.
In case the input is a frame the command will first check if the descriptor HISTOGRAM exits. If so, these data will be plotted. If the descriptor is absent the command STAT/IMAGE will be executed first.
The histogram determination of a table column makes use of the command READ/HISTOGRAM. The histogram data for tables is stored in two descriptors of the table: TCLAS001 and TFREQ001.

**See also:** PLOT/HISTOGRAM, PRINT/HISTOGRAM, COMPUTE/HISTOGRAM, READ/HISTOGRAM, STAT/IMAGE

**Examples:** `OVERPLOT/HISTOGRAM myimage ? LOG`
This will overplot the descriptor HISTOGRAM of the image myimage.bdf with a logarithmic scale in y.

`OVERPLOT/HISTOGRAM mytable :VELOCITY .5 1,8 NN ? 1,.5,90`
This will plot the column labelled :VELOCITY from table mytable.tbl with a y offset of .5 units; the bin size will be 1 and the minimum value will be 8. Excess bin are not plotted and the histogram is of the bar type and filled with vertical lines.

## OVERPLOT/KEYWORD

*core* 22-SEP-1988 RHW

**Purpose:** Overplot the contents of a keyword

**Subject:** Graphics, keywords

**Syntax:** `OVERPLOT/KEYWORD [key_name] [start,end] [offset]`

| | |
|---|---|
| `key_name` | name of the keyword; default is OUTPUTR |
| `start,end` | first and last data point to be overplotted; end can be smaller than start; defaults are the first and the last valid data points in the keyword |
| `offset` | offset value in the y direction (world coordinate units); default 0.0 |

**Note:** none

**See also:** PLOT/KEYWORD, WRITE/KEYWORD, SET/GRAPHICS, ASSIGN/GRAPHICS

**Examples:** `OVERPLOT/KEYWORD YOURKEY ? 5.0`
This will overplot all the data points of the keyword YOURKEY with an offset of 5.0.

# OVERPLOT/LINE

## OVERPLOT/LINE *core* 17-SEP-1988 RHW

**Purpose:** Overplot a line on a graphics device

**Subject:** Graphics, lines

**Syntax:** `OVERPLOT/LINE [ltype] [x_sta,y_sta[,mm] [x_end,y_end]]`

`ltype` number: 0 – no line at all; 1 – solid; 2 – dotted; 3 – short dash; 4 – dash - dot; 5 – long dash; 6 – dash - dot - dot; default is 1

`x_sta,y_sta[,mm]` ing world coordinate or 'C' for cursor input; default is 'C'. If the flag ",mm" is given the (start) position will be x_sta mm from the left y_axis boundary and y_sta mm for the bottom x axis boundary. Else, the position is interpreted as a world coordinate position. In case NO coordinate pair is entered the graphics cursor will appear, allowing you to enter the begin and end points with the left mouse button. Exit with the right mouse button.

`x_end,y_end[,mm]` world coordinate; for format see above. This input is only needed if x_sta,x_end are specified. The end coordinate are considered to be expressed in the same coordinate system as the start coordinate (i.e. in mm or in world coordinates).

**Note:** The command allows to overplot a line outside the graphics frame. In such a case, if one wants to input start and end coordinates an extrapolation of the x- and/or y-axis should be applied.
The command allows you to re-iterate: as long as the exit button isn't pressed you can choose another position of the line with the left button.

**See also:** SET/GRAPHICS, LABEL/GRAPHICS, OVERPLOT/SYMBOL, OVERPLOT/ERROR, OVERPLOT/GRID

**Examples:** `PLOT/ROW myframe @125 @100,@440`
OVERPLOT/LINE 3 ? Plot a line of a frame and overplot a line in line type 3 using the graphic cursor.

`PLOT/ROW myframe @125 @100,@440`
OVERPLOT/LINE 6 250,50 350,50 As in the previous example but with the start and end coordinates given.

## OVERPLOT/ROW *core* 04-SEP-1991 RHW

**Purpose:** Overplot a row (line) of a frame on a graphic device

**Subject:** Graphics, frames

**Syntax:** `OVERPLOT/ROW frame [y_coord] [x_sta,x_end] [offset] [l_type]`

| | |
|---|---|
| `frame` | image file name |
| `y_coord` | row (line) number or world coordinate in the frame; default the first row. |
| `x_sta,x_end` | can be: (a) first and last pixel coordinate; (b) start, end in world coordinates on the line. Default is either the manual setting done by the command SET/GRAPHICS (if present), or the whole line. |
| `offset` | offset in pixel intensity units; default 0.0 |
| `l_type` | line type to be used; default is the currently enabled line type (use SHOW/GRAPHICS). |

**Note:** none

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, PLOT/ROW, PLOT/COL, OVERPLOT/COL

**Examples:** `OVERPLOT/ROW myframe @256 @10,@100 100`
This command overplots from image myframe.bdf line 256, pixels 10 to 100 with an offset of 100 units.

# OVERPLOT/SYMBOL

## OVERPLOT/SYMBOL *core* 17-SEP-1988 RHW

**Purpose:** Overplot a symbol

**Subject:** Graphics, symbols

**Syntax:** `OVERPLOT/SYMBOL [s_type] [x_pos,y_pos[,mm]] [s_size]`

`s_type` symbol type. The following symbols are possible: 0 means no symbol at all; 1 – dot; 2 – circle; 3 – square; 4 – triangle; 5 – cross (+); 6 – cross (x); 7 – asterisk; 8 – star; 9 – crossed square (x); 10 – crossed square (x); 11 – losange; 12 – hor. bar; 13 – vert. bar; 14 – right arrow; 15 – arrow up; 16 – left arrow; 17 – arrow down; 18 – filled exagon; 19 – filled square; 20 – filled triangle; 21 – filled lozenge; default 5 (cross)

`x_pos,y_pos[,mm]` position of the text in the plot. If the flag ",mm" is given the label will be drawn x_pos mm. from the left y-axis boundary and y_pos mm. from the bottom y axis boundary. Else, the position is interpreted as a world coordinate position. In case NO coordinate pair is entered the graphics cursor will appear, allowing you to enter the position with the left mouse button. Exit with the right mouse button.

`s_size` multiplication factor for size of the symbol; default 1.0

**Note:** The command allows to overplot a symbol outside the graphics frame. In such a case, if one wants to input coordinates an extrapolation of the x- and/or y-axis should be applied.
The command allows you to re-iterate: as long as the exit button isn't pressed you can choose another position of the symbol with the left button.

**See also:** PLOT/TABLE, PLOT/ROW, PLOT/COLUMN, SET/GRAPHICS, SHOW/GRAPHICS

**Examples:**

```
PLOT/ROW myframe @125 @100,@440

OVER/SYM 9 4010,53 2.0
```

## OVERPLOT/TABLE *core* 01-OCT-1993 RvH

**Purpose:** Overplot table data on selected plotting device

**Syntax:** `OVERPLOT/TABLE table [plane1] [plane2] [x_sc,y_sc[,x_off,y_off]]`

`[symbols] [lines] [flag_dir]`

| | |
|---|---|
| `table` | name of table file |
| `plane1` | vector or plane in abscissa (x-axis); default is sequence number, however see the note below. |
| `plane2` | vector or plane in ordinate (y-axis); default is sequence number, however see the note below. |
| `symbols` | symbol types to be used in the plot; input can be given as s_start:s_end:s_incr or s_nr1,s_nr2,s_nr3, ... or any combination of these two possibilities separated by a comma. Default is the symbol type set by SET/GRAP. Data points on different lines in the extracted planes will be presented with different symbols if more than one symbol type is given. The program will cycle though the given symbols if the number of lines exceed the number of given symbols. |
| `lines` | line types to be used in the plot; input can be given as l_start:l_end:l_incr or l_nr1,l_nr2,l_nr3, ... or any combination of these two possibilities separated by a comma. PLOT/TABLE will draw lines if STYPE is equal to zero or if you specify here one or more line type to be used in the plot. Data points on the first axis of a plane will be connected with a line of given type, and if the number of lines exceed the number of given line types the program will cycle through the line types. |
| `flag_dir` | flag_dir specifies the way both planes are read. Two values can be given: "D"efault and "O"pposite.<br>- a plane along a column in the depth direction is by default stored column by column (thus the first axis is along the columns, and each column can be represented by a symbol and/or line type).<br>- a plane along a row in the depth direction is by default stored array by array.<br>- a plane at a certain depth is stored column by column<br>Note that this flag also affects the way a vector is treated! |

# OVERPLOT/TABLE

**Note:** The new OVERPLOT/TABLE command is able to display data from a 3-D table. The syntax to define planes and vectors in a 3-D table is still experimental!!!
A 3-D table has 3 axes: column, row and depth (or array). The preliminary syntax allows you to define the following planes:
#n[i..j] : a plane along column "n", from depth "i" to "j",
@r#n..m[i..j] : a plane along row "r", from column "n" to "m" and from depth "i" to "j",
[i]#n..m : a plane along depth "i", from column "n" to "m".
It is also possible to refer to a column by its name ":name". The default for the range in depth or columns are all selected elements along the depth or column axis. To specify one element one may type [i] or #n. A range of rows can only be selected by using the command SELECT/TABLE.

When the user gives a "?" for one of the two planes the values in the other plane will plotted against its sequence number. Only one of the plane input parameters can be defaulted to sequence number. So, 'table plane1 ?' and 'table ? plane2' are both valid input parameter strings, 'table ? ?' is not.

The program allows to plot data extracted from planes with different orientations against each other. But the number of elements along the first axis of the planes have to be equal. Remember that you can define the first axis by using "flag_dir".

It is possible to plot a vector against a plane or the other way around:
plane1 = @3[4] (: a vector along row 3 at array element 4) and
plane2 = @6 (: a plane with all the data values at row 6).
The program will apply the same restrictions to the first plane as put on the second plane, ONLY if both planes have the same orientation AND if the first plane is defaulted to all elements in a certain direction, while the second is not.
For example: if plane1 = #2 and plane2 = #3[2..5] then the program will only extract array elements 2 to 5 from both planes. If you do not want this you will have to define first plane2 and plane1 as the next parameter or explicitly define the first plane from the first to the last element.

The 2-D version of OVERPLOT/TABLE is rewritten in such a way that the old parameter list is still valid if your input table is 2-D. The program will treat a column as a plane with one array element at depth = 1. But you can also define a row and plot it against an other row, column or sequence number.

The functions LOG and LN, decimal and natural logs respectively, can be applied to the data to be plotted.

Be aware that in case you plan to connect the data points with a line (see SET/GRAPHICS) the data points will be connected in the same order as they appear in the table. In order to sort the data first (either in decreasing or increasing order) use the sorting command SORT/TABLE.

**See also:** PLOT/TABLE, PLOT/ERROR, TUTORIAL/GRAPH,SET/GRAPHICS, SHOW/GRAPHICS

**Examples:**
```
OVERPLOT/TAB 2-Dtable :MAGNITUDE LOG(:TEMPERATURE)
```
Overplot column :MAGNITUDE versus the logarithmic value of column :TEMPERATURE (base 10) of table '2-Dtable'.
```
OVERPLOT/TAB 3-Dtable LN(#3[1]) ?  3:6:1,10 ?  Def,Op
```

Will do the same as the previous example. But it will plot the first array element of a plane along column 3 against sequence number.

```
OVERPLOT/TAB 3-Dtable #3 #5 3:6:1,10 1,4
```

Plot the values found in a plane along column 3 against column 5. The data will be presented according to the following scheme:
array element: 1 2 3 4 5 6 7 8 etc.
line type : 1 4 1 4 1 4 1 4 ...
symbol type : 3 4 5 6 10 3 4 5 ...

```
OVERPLOT/TAB 3-Dtable @3:name @5 3:6:1,10 1,4
```

Take a vector at row 3, column "name" as abscissa and the plane along row 5 in the depth direction as ordinate. Plotted with the same line and symbol types as in the previous example. The vector and plane are both read (by default) array by array, thus there are no dimensional problems.

```
OVERPLOT/TAB 3-Dtable [3]#1..4 #2[1..4] 3:6:1,10 1,4
```

Take plane along depth 3, from column 1 to 4 as abscissa and the plane along column 2, from depth 1 to 4 as ordinate. Both planes are read column by column so four lines will be plotted with resp. line type 1, 4, 1, 4.

# OVERPLOT/VECTOR

## OVERPLOT/VECTOR *core* 26-AUG-1991 RHW

**Purpose:** Overplot vector map from two 2-dim. images with smoothing option

**Syntax:** `OVERPL/VECTOR fram_a fram_b [coord_str] [scale_r] [pos_range] [sm_par] [head]`

| | |
|---|---|
| `fram_a` | input frame containing the intensities |
| `fram_b` | input frame containing the position angles |
| `coord_str` | area to be plotted in standard MIDAS notation; see MIDAS Users Guide Chapter 3. Default is either the manual setting done with by the SET/GRAPHICS command (is present), or the whole area. |
| `scale_r` | scale of the vector length in intensity units/mm. By default the command computes the average of the minimimum and maximum cut values in the descriptor of frame_a; this average intensity will correspond with a vector length of 10 mm. |
| `pos_range` | range of positions angle to be plotted in degrees. Default is 0,360. The position angle is defined using the standard astronomical convention: 0 is pointing to the top, and the angle increases counter-clockwise. |
| `sm_par` | smoothing parameter i.e. box width in pixels; default is 1 |
| `head` | head parameter. If the vector should have an arrow at its end this parameters should be 1, else give a 0. Default is 1, i.e. the arrow is drawn. |

**Note:** The coordinates for the area are optional. If one specifies a "?", the area will be take for the manual frame settings for the x- and y-axis. If "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already. To pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.
The maximum area that can be plotted in 512 by 512 pixels.

**See also:** PLOT/VECTOR, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES, PLOT/CONTOUR, OVERPLOT/CONTOUR, PLOT/GRAY, OVERPLOT/GRAY

**Examples:** `PLOT/AXES 100,400,100,50 100,400,100,25 ?`
OVERPL/VECT spir_i spir_p [@281,@281:@320,@320] 0.5 ? 3 Overplot the polarisation vector using the intensities from spir_i and the position angle information form spir_p. Use a scale of 0.5 units/mm for the size of the vectors (i.e. an intensity of 10 pixel units will make the vector 20 mm). Apply a box smooth over 3x3 pixels.

## PLAYBACK/FILE *core* 23-DEC-1993 KB

**Purpose:** Playback MIDAS commands from an ASCII file.

**Syntax:** `PLAYBACK/FILE name`

| | |
|---|---|
| `name` | name of ASCII file containing MIDAS commands;<br>file type ".log" is appended, if no file type explicitely given |

**See also:** PLAYBACK/LOGFILE

**Note:** PLAYBACK/FILE can be only entered interactively!
Nesting of playback files is not possible (i.e. you cannot playback a file which contains itself the command PLAYBACK).
Do not mix up the command PLAYBACK/FILE with @, @@:
PLAYBACK/FILE reads a file line by line and feeds them to MIDAS just like terminal input. That's why a playback file can only contain interactive MIDAS commands, no commands like IF or GOTO are possible.

**Examples:** `PLAYBACK/FILE MID_WORK:myfile.dat`
Feed the lines contained in the ASCII file 'myfile.dat' located in MID_WORK as input to MIDAS.

## PLAYBACK/LOGFILE *core* 02-OCT-1991 KB

**Purpose:** Playback commands from a MIDAS logfile.

**Syntax:** `PLAYBACK/LOGFILE file`

| | |
|---|---|
| `file` | a logfile (or parts of it) of a previous MIDAS session;<br>file type ".log" is appended, if no file type explicitely given |

**See also:** LOG/ON, LOG/OFF, PLAYBACK/FILE

**Note:** Only MIDAS commands, i.e. lines which begin with "Midas xyz>" are taken into account, all other file records are ignored.

**Examples:** `PLAYBACK/LOG platano`
Redo the Midas commands stored in ASCII file 'platano.log'.

# PLOT/AXES

## PLOT/AXES *core* 21-sep-1990 RHW

**Purpose:** Plot a coordinate box with large and small tickmarks

**Syntax:** `PLOT/AXES [x_axis_spec] [y_axis_spec] [x_sc,y_sc[,x_off,y_off]]`

`[x_lab] [y_lab]`

`PLOT/AXES [coord_str] [x_lab] [y_lab]`

`x_axis_spec` x-axis specification: start value, end value, distance between the big tickmarks, distance between the small tickmarks. The start value can be smaller than the end_value. If distance between small tickmarks is less than 0, a logarithmic axis will be plotted, running from 10**start_value to 10**end_value. At least the start and end values should be given. If large tickmark distance fails a (hopefully) sensible default will be used. If small tickmark distance fails no small tickmarks will be plotted. Default: use the manual setting (by the SET/GRAPHICS command); if not present, the default is a linear axis from 0.0 to 1.0.

`y_axis_spec` specfication for the y-axis; see above.

`coord_str` area in the displayed frame in the standard MIDAS notation. This option only works on workstations, if the display window has been assigned as the plot device and a frame has been loaded! The default is a frame box around the whole frame.

`x_sc,y_sc,x_off,y_off`
scale or size in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`x_lab` label for the x-axis; default is no label

`y_lab` label for the y-axis; default is no label

**Note:** The command can obtain its input either from the manual settings by the SET/GRAP command, or from input on the command line. In case a manual setting for one or both axis is used and the large tickmark distance fails, no large tickmarks will be drawn (only the start and end positions of the axis). In case of automatic plotting the command tries to find (hopefully) sensible positions for the large tick marks. Similar behaviour can be found for all other plot commands.
The command can draw axes on all graphic output devices, including the display window in workstations. Depending on the assigned graphic device (with ASSIGN/GRAPHICS) the command decides in which mode it will run.
In case the output graphics device is the graphics window, terminal or a hardcopy device the command will always run in the first mode.
If the assigned output graphics device is the display window AND if the first input parameter is given as a MIDAS coordinate string (i.e. starting with '[') the command will run in the second mode (see below) and will be drawn around the LOADED frame. However, with the display window assigned, if the first parameter is not a MIDAS coordinate string the command will run in the first mode. In the second mode, if (part of) the frame comes close to display window boundary part of the coordinate frame will fall outside the window. In that case one can either center the frame, or give a somewhat smaller area to draw the coordiates around.

**See also:** ASSIGN/GRAPHICS, SET/GRAPHICS, SHOW/GRAPHICS, OVERPLOT/AXES

**Examples:** `PLOT/AXES 0,100,25,5 1,3,1,-1 0.8,0.1,25,10 "V_Hel" "Flux"`
Plot a box with a linear scale in the x and a logarithmic one in the y direction; use the default scales in both x and y direction. The box will be put 150 mm from the left border of the graphics device

`LOAD/IMAGE spiral`
Load a frame

`ASSIGN/GRAPHICS display`
Assign the display window as the graphics device

`PLOT/AXES`
Since the display window is the assigned graphic device the command will try to draw a box around the displayed frame.

`PLOT/AXES [@10,@10:@210,@305] "Right Ascension" "Declination"`
Plot a coordinate frame around the displayed frame.

`PLOT/AXES 0,1,.25,.5 0,1 0,0,30,50 ? ?`
A frame will be drawn on the display window with an offset of 30 mm in x and 50 mm in y with respect to the graphic device origin (in this case the display window).

# PLOT/COLUMN

## PLOT/COLUMN *core* 04-SEP-1991 RHW

**Purpose:** Plot a column of an image on the selected plotting device

**Syntax:** `PLOT/COLUMN frame [x_coord] [y_sta,y_end] [x_sc,y_sc[,x_off,y_off]]`

| | |
|---|---|
| `frame` | image file name |
| `x_coord` | column number or world coordinate in the frame; default the first column |
| `y_sta,y_end` | can be: (a) first and last pixel in the column; (b) start, end in world coordinates on the column. Default is either the manual setting done by the command SET/GRAPHICS (if present), or the whole column. |
| `x_sc,y_sc,x_off,y_off` | scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.<br>x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels. |

**Note:** none

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, OVERPLOT/COLUMN, PLOT/ROW, OVERPLOT/ROW

**Examples:**

```
PLOT/COLUMN myframe @256 @10,@100
```

This command will plot from image myframe.bdf column 256, pixels 10 to 100.

```
PLOT/COLUMN myframe 256 100,10 13.6,5,20,30
```

This command plots from image myframe.bdf the column corresponding to the x coordinate 256, in the range from 100 to 10 (all world coordinates) with a negative increment. The scales of the plot are along the x axis: 13.6 world coord. unit/mm.; in y: 13.6 intensity units/mm. The plot will be drawn at 20 mm from the left edge and 30 mm from the botom edge of the graphic device.

```
PLOT/COLUMN myframe 256 100,10 13.6,-100
```

This command plots from image myframe.bdf the column corresponding to the x coordinate 256, in the range from 100 to 10 (all in world coordinates) with a negative increment. The scales of the plot are along the x axis again 13.6 world coord. unit/mm.; the plot will be 10 cm high.

## PLOT/CONTOUR *core* 09-JUN-1987 RHW

**Purpose:** Plot contour map of 2-dim. image (with a smoothing option).

**Syntax:** `PLOT/CONTOUR frame [coord_str] [x_sc,y_sc[,x_off,y_off]]`

`[contours] [c_type] [sm_par]`

`frame` name of image file

`coord_str` area to be plotted in standard MIDAS notation; see MIDAS Users Guide Chapter 3. Default is either the manual setting done by the SET/GRAPHICS command, or the whole area.

`x_sc,y_sc,x_off,y_off` scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`contours` contour values; input can be given as cstart:cend:cincr or cnt1,cnt2,cnt3 or any combination of these two possibilities separated by a comma.

`c_type` NEG, ODD, or LTYPE: determines the line type(s) used. NEG will draw negative contours dashed and positive ones solid; ODD will draw odd contours dashed and even ones solid; If LTYPE is specified contours will be drawn with the line type specified with SET/GRAPHICS. Default is ODD

`sm_par` smoothing parameter i.e. the number of pixels "around" the central pixel which define the neighbourhood of that pixel.
Total number of pixels is then: (2*sm_par + 1)**2 Only integer numbers are allowed. Default is 0

**Note:** The coordinates for the area are optional. If one specifies a "?", and the manual frame setting(s) for the x- and y-axis are used these settings will be used; else the whole area will be taken. If a "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already.
For the DEANZA to pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.
For XWindows use the mouse to move the cursor, the leftmost mouse button or the RETURN key is the ENTER button, the second left mouse button serves as EXIT button. The cursor cross may not be visible at first. Move the mouse to the lower left corner of the graphics window to grab the cursor.
The size of the plot area (area within the rectangular cursor) can be be changed by fixing one of the two cursor corners (DEANZA) or by using the arrow keys on the keyboard (XWindows).

**See also:** OVERPLOT/CONTOUR, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES, PLOT/GRAY, OVERPLOT/GRAY, PLOT/VECTOR, OVERPLOT/VECTOR

**Examples:** `PLOT/CONTOUR spiral [@281,@281:@320,@320] ? 1:5:0.25,6,7,8 ? 1`
Plot a contour plot of the frame spiral.bdf with default scale; the contour are 1 to 5 in steps of .25 and the contours 6, 7, and 8. A 3*3 box smooth is done on the data before the plot is made. Use the default line type.

# PLOT/DESCRIPTOR

## PLOT/DESCRIPTOR *core* 09-JUN-1987 RHW

**Purpose:** Plot descriptor on the selected plotting device

**Syntax:** `PLOT/DESCRIPTOR frame [descr] [start,end] [x_sc,y_sc[,x_off,y_off]]`

`frame` name of data frame

`descr` name of the descriptor. The decriptor should be of the type integer, real or double precision; default is HISTOGRAM

`start,end` first and last data point to be plotted; end can be smaller than start; defaults are the first and the last valid data points in the descriptor.

`x_sc,y_sc,x_off,y_off` scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

**Note:** The command is useful to plot data stored in a descriptor. The maximum number of data points in the descriptor that can be plotted is 10000. With more data points one should use additional overplot commands.

**See also:** WRITE/DESCIPTOR, READ/DESCRIPTOR, COPY/DD, COPY/DK, COPY/KD

**Examples:** `PLOT/DESCRIPTOR myframe histogram 100,125`
This will plot the descriptor histogram associated with the image myframe.bdf. Only the elements 100 to 125 will be plotted. The descriptor must exist. In this case, it should have been created previously by the command STATISTICS/IMAGE myframe.

PLOT/GRAY *core* 06-AUG-1987 RHW

**Purpose:** Plot gray scale map of 2-dim. image (with a smoothing option)

**Syntax:** `PLOT/GRAY frame [coord_str] [x_sc,y_sc[,x_off,y_off]] [gray_lev]`

`[sm_par] [gray_ness] [gray_opt]`

`frame` name of image file

`coord_str` area to be plotted in standard MIDAS notation; see MIDAS Users Guide Volume A chapter 6. Default is the manual setting done with the command SET/GRAPHICS (if present), or the whole area.

`x_sc,y_sc,x_off,y_off` scaling factor in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`gray_lev` gray levels; in case of a continueous gray scale plot two values are required (lev1,lev2); in case of stepwise increment of the gray scale plot input can be given as cstart:cend:cincr or lev1,lev2,lev3 or any combination of these two possibilities separated by a comma; defaults cut levels are the values stored in the decriptor LHCUTS.

`sm_par` smoothing parameter i.e. the number of pixels "around" the central pixel which define the neighbourhood of that pixel.
Total number of pixels is then: (2*sm_par + 1)**2 Only integer numbers are allowed. Default is 0

`gray_ness` grayness parameter between 0.0 and 1.0; the value 1 gives the maximum blackness; 0 means white; default is 1.0

`gray_opt` plot options to be specified as option1,option2,...; possible options are:
LOG: logarithmicaly increasing grayness;
LIN: linear increasing grayness (default);
STEP: stepwise gray scales;
CONT: contineously increasing gray scales;
NEG: change sign of data points;
POS: leave signs unchanged (default);
ABS: take absolute values;
CUT: intensity levels above the highest contour level will be white. The default is that these levels will have a grayness corresponding the highest contour level.

**Note:** The coordinates for the area are optional. If one specifies a "?", the area will be take for the manual frame settings for the x- and y-axis. If a "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already. To pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.

**See also:** OVERPLOT/GRAY, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES, PLOT/CONTOUR, OVERPLOT/CONTOUR, PLOT/VECTOR, OVERPLOT/VECTOR

**Examples:** `PLOT/GRAY spiral [@281,@281:@320,@320] ? 1,5`

# PLOT/GRAY

Plot a gray scale plot of the frame spiral.bdf with default scales; the gray levels will be continuous in the range from pixel values 1 to 5.

```
PLOT/GRAY spiral [@281,@281:@320,@320] 25.,25.  1:5 1 1.  ABS,LOG
```

Same as the previous example but a logarithmic scaling is done to the absolute values of the data points. A 3*3 box smooth is done on the data before the plot is made.

```
PLOT/GRAY spiral C ? 1:5:0.5 ?  .75 STEP
```

In this case the gray levels will increase stepwise. To get the plot less black the grayness parameter is set to 0.75.

PLOT/HISTOGRAM *core* 10-JUN-1987 RHW

**Purpose:** Plot a histogram of a column in a table or plot the histogram of given image

**Syntax:** `PLOT/HISTOGRAM tab col [x_sc,y_sc[,x_off,y_off]] [bin[,min[,max]]]`

`[exc] [log] [opt]`

`or`

`PLOT/HISTOGRAM frame [x_sc,y_sc[,x_off,y_off]] [exs] [log] [opt]`

| | |
|---|---|
| `tab` | name of table file |
| `col` | column reference |
| `frame` | name of image frame |
| `x_sc,y_sc,x_off,y_off` | scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.<br>x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels. |
| `bin,min,max` | bin size, minimum, and maximum values to be included in the histogram. Default bin size depends on the actual dynamic range of the data values; default minimum is the minimum of the column values; similar for maximum. |
| `exc` | plot low and high excess; default YY |
| `log` | flag for logarithmic scale; can be LIN (default) LOG or LN. |
| `opt` | histogram type, hashing spacing and hashing angle; default: standard staircase plot, no hashing done.<br>The histogram type is determined by an integer number: 0: simple staircase;<br>1: staircase steps joining the x axis;<br>=> 2: data points will joint the x axis with boxes which are determined by the number (2 = zero width)<br><br>In case the hashing spacing is given a small number (e.g. 0.01) the histogram will be filled completely. A reasonable number is 1. |

**Note:** In case the second parameter is defaulted, the command assumes a frame as input.
In case the input is a frame the command will first check if the descriptor HISTOGRAM exits. If so, these data will be plotted. If the descriptor is absent the command STATISTICS/IMAGE will be executed first.
The histogram determination of a table column makes use of the command READ/HISTOGRAM. The histogram data for tables is stored in two descriptors of the table: TCLAS001 and TFREQ001.

**See also:** PRINT/HISTOGRAM, COMPUTE/HISTOGRAM, READ/HISTOGRAM
STATISTICS/IMAGE, STATISTICS/TABLE

**Examples:** `PLOT/HISTOGRAM myimage ? NN LOG`
This will plot the descriptor HISTOGRAM of the image 'myimage.bdf' with a logarithmic scale in y; the plot will fill the plot area of the device. Does not plot the excess bins of the histogram if existing.

# PLOT/KEYWORD

```
PLOT/HISTOGRAM mytable :VELOCITY 0.5,12.34 1,8 ?  ?  1,1,45
```
This will plot the column labelled :VELOCITY from table 'mytable.tbl' with a x-scale of 0.5 units/mm and in y 12.34 units/mm; the bin size will be 1 and the minimum value will be 8. Excess bins will be plotted, and the histogram will be of bar type and will be hashed (angle 45 deg).

---

PLOT/KEYWORD *core* 22-SEP-1988 RHW

**Purpose:** Plot the contents of a keyword on a graphics device

**Syntax:** `PLOT/KEYWORD [key_name] [start,end] [x_sc,y_sc[,x_off,y_off]]`

`key_name` name of the keyword; default OUTPUTR

`start,end` first and last data point to be plotted; end can be smaller than start; defaults are the first and the last valid data points in the keyword.

`x_sc,y_sc,x_off,y_off` scaling in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

**Note:** none

**See also:** OVERPLOT/KEYWORD, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES,

**Examples:** `PLOT/KEYWORD MYKEY 5,25`
Plot the keyword MYKEY from the 5th to the 25th data point.

## PLOT/PERSPECTIVE *core* 25-APR-1994 RvH

**Purpose:** Plot 3 dim. representation of a 2 dim. image, with smoothing option

**Syntax:** `PLOT/PERSPECTIVE frame [coord_str] [alt,azi]] [scal,offs] [sm_par]`

`[xy_flag]`

| | |
|---|---|
| `frame` | name of image file |
| `coord_str` | area to be plotted in standard MIDAS notation; see MIDAS Users Guide Chapter 3. Default is the manual setting done with the command SET/GRAPHICS (if present), or the whole area (see note below) |
| `alt,azi` | viewing angles altitude (angle above the plane of the plot: alt=0 is edge-on; alt=90 is face-on) and azimuth (angle in the plane: azi=0 for the x-axis horizontally at the lower part of the plot; azi=90 for the y-axis horizontally at the lower part). Default 45,30 |
| `scal,offs` | scale and offset parameters. The scale controls the height coordinate (ZAXIS) with respect to the graphics area. The plot will (by default) fill the graphics area (-> scale = 1). Scale = 0.5 will make the zaxis half of its default size. The offset control controls the offset of the data in the z-direction (NOT of the axes). The default is 0. |
| `sm_par` | smoothing parameter i.e. the number of pixels "around" the central pixel which define the neighbourhood of that pixel.<br>Total number of pixels is then: (2*sm_par + 1)**2 Only integer numbers are allowed. Default is 0 |
| `xy_flag` | flag to indicate which lines should be drawn: "X" the x-direction only, "Y" the y-direction only, and "XY" both the x- and y-direction (default) |

**Note:** The coordinates for the area are optional. If one specifies a "?", the area will be take for the manual frame settings for the x- and y-axis. If a "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already. To pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.
The maximum area that can be plotted is 512 by 512 pixels.
The X, Y and Z axis can be defined with SET/GRAPHICS
In plot mode 1 and 2 (see SET/GRAPHICS) a coordinate frame is plotted around the 3-dim view at the level of the minimum and maximum found in the frame used.
PLOT/PERSEPTIVE has some difficulties drawing almost vertical lines (bottom to top), i.e. the surface is often not closed very nicely. By plotting only the horizontal lines (using the xy_flag) you will get a nice results. The xy_flag is also very useful if you are plotting more then 100 lines.

**See also:** PLOT/CONTOUR, SET/GRAPHICS, SHOW/GRAPHICS

**Examples:** `plot/perspec spiral [@200,@200:@400,@400]`
Make a perspective plot of the frame spiral, using the default viewing angles.

`plot/perspec spiral [@200,@200:@400,@400] 25,230 .2,-1 2 x`
The same image as before but the image will be seen under different viewing angles, the intensity is scaled down by a factor 5, with an offset of -1, the data is smoothed by taking a box of 5 pixels around each pixel, and only the x lines are shown.

# PLOT/ROW

PLOT/ROW *core* 04-SEP-1991 RHW

**Purpose:** Plot a row (line) of an image on the selected plotting device

**Syntax:** `PLOT/ROW frame [y_coord] [x_sta,x_end] [x_sc,y_sc[,x_off,y_off]]`

`frame` image file name

`y_coord` row (line) number or world coordinate in the frame; default the first row.

`x_sta,x_end` can be: (a) first and last pixel coordinate; (b) start, end in world coordinates on the line. Default is either any manual setting done by the command SET/GRAPHICS, or the whole line.

`x_sc,y_sc,x_off,y_off` scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y-sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

**Note:** The y-axis is labeled with the string "Pixel value(CUNIT(1:16))" and the x-axis with "Position (CUNIT(17:32)). Where e.g. CUNIT(17:32) means the contents of the descriptor CUNIT (from char. 17 until char. 32) of the plotted frame.

**See also:** SET/GRAPHICS, SHOW/GRAPHICS, OVERPLOT/ROW, PLOT/COL, OVERPLOT/COL

**Examples:** `PLOT/ROW myframe @256 @10,@100`
This command will plot from image myframe.bdf line 256, pixels 10 to 100.

`PLOT/ROW myframe 256 100,10 13.6,5`
This command will plot from image myframe.bdf the line corresponding to the y coordinate 256, in the range from 100 to 10 in world coordinates with a negative increment. The scales are in x: 13.6 world coord. unit/mm.; in y: 13.6 intensity units/mm.

`PLOT/ROW myframe 256 100,10 13.6,-100`
This command will plot from image myframe.bdf the line corresponding to the y coordinate 256, in the range from 100 to 10 in world coordinates with a negative increment. The scale along the x axis is again 13.6 world coord. unit/mm.; the plot will be 10 cm high.

PLOT/TABLE *core* 01-OCT-1993 RvH

**Purpose:** Plot table data on selected plotting device

**Syntax:** `PLOT/TABLE table [plane1] [plane2] [x_sc,y_sc[,x_off,y_off]]`

`[symbols] [lines] [flag_dir]`

`table` name of table file

`plane1` vector or plane in abscissa (x-axis); default is sequence number, however see the note below.

`plane2` vector or plane in ordinate (y-axis); default is sequence number, however see the note below.

`x_sc,y_sc,x_off,y_off` scaling in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y_sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made *_sc mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`symbols` symbol types to be used in the plot; input can be given as s_start:s_end:s_incr or s_nr1,s_nr2,s_nr3, ... or any combination of these two possibilities separated by a comma. Default is the symbol type set by SET/GRAP. Data points on different lines in the extracted planes will be presented with different symbols if more than one symbol type is given. The program will cycle though the given symbols if the number of lines exceed the number of given symbols.

`lines` line types to be used in the plot; input can be given as l_start:l_end:l_incr or l_nr1,l_nr2,l_nr3, ... or any combination of these two possibilities separated by a comma. PLOT/TABLE will draw lines if STYPE is equal to zero or if you specify here one or more line type to be used in the plot. Data points on the first axis of a plane will be connected with a line of given type, and if the number of lines exceed the number of given line types the program will cycle through the line types.

`flag_dir` flag_dir specifies the way both planes are read. Two values can be given: "D"efault and "O"pposite.
- a plane along a column in the depth direction is by default stored column by column (thus the first axis is along the columns, and each column can be represented by a symbol and/or line type).
- a plane along a row in the depth direction is by default stored array by array.
- a plane at a certain depth is stored column by column
Note that this flag also affects the way a vector is treated!

# PLOT/TABLE

**Note:** The new PLOT/TABLE command is able to display data from a 3-D table. The syntax to define planes and vectors in a 3-D table is still experimental!!!
A 3-D table has 3 axes: column, row and depth (or array). The preliminary syntax allows you to define the following planes:
#n[i..j] : a plane along column "n", from depth "i" to "j",
@r#n..m[i..j] : a plane along row "r", from column "n" to "m" and from depth "i" to "j",
[i]#n..m : a plane along depth "i", from column "n" to "m".
It is also possible to refer to a column by its name ":name". The default for the range in depth or columns are all selected elements along the depth or column axis. To specify one element one may type [i] or #n. A range of rows can only be selected by using the command SELECT/TABLE.

When the user gives a "?" for one of the two planes the values in the other plane will plotted against its sequence number. Only one of the plane input parameters can be defaulted to sequence number. So, 'table plane1 ?' and 'table ? plane2' are both valid input parameter strings, 'table ? ?' is not.

The program allows to plot data extracted from planes with different orientations against each other. But the number of elements along the first axis of the planes have to be equal. Remember that you can define the first axis by using "flag_dir".

It is possible to plot a vector against a plane or the other way around:
plane1 = @3[4] (: a vector along row 3 at array element 4) and
plane2 = @6 (: a plane with all the data values at row 6).
The program will apply the same restrictions to the first plane as put on the second plane, ONLY if both planes have the same orientation AND if the first plane is defaulted to all elements in a certain direction, while the second is not.
For example: if plane1 = #2 and plane2 = #3[2..5] then the program will only extract array elements 2 to 5 from both planes. If you do not want this you will have to define first plane2 and plane1 as the next parameter or explicitly define the first plane from the first to the last element.

The 2-D version of PLOT/TABLE is rewritten in such a way that the old parameter list is still valid if your input table is 2-D. The program will treat a column as a plane with one array element at depth = 1. But you can also define a row and plot it against an other row, column or sequence number.

The functions LOG and LN, decimal and natural logs respectively, can be applied to the data to be plotted.

Be aware that in case you plan to connect the data points with a line (see SET/GRAPHICS) the data points will be connected in the same order as they appear in the table. In order to sort the data first (either in decreasing or increasing order) use the sorting command SORT/TABLE.

**See also:** OVERPLOT/TABLE, OVERPLOT/ERROR, SET/GRAPHICS, SHOW/GRAPHICS

**Examples:**
```
PLOT/TAB 2-Dtable :MAGNITUDE LOG(:TEMPERATURE) 2,5
```
Plot column :MAGNITUDE versus the logarithmic value of column :TEMPERATURE (base 10) of table '2-Dtable'; to get the right size of the plot use scales 2 (in X) and 5 (in Y) unit/mm.
```
PLOT/TAB 2-Dtable LN(#3) ?  0.5,1.234 3:6:1,10 ?  Def,Op
```

Plot the logarithmic value (base e) of column 3 in table '2-Dtable' against sequence number. Use x and y scale 0.5 and 1.234 units/mm respectively. The data points are plotted with different symbols (3,4,5,6,10). If column 3 contains more than 5 values then value 6 will be presented with symbol 3, etc

```
PLOT/TAB 3-Dtable LN(#3[1]) ?  0.5,1.234 3:6:1,10 ?  Def,Op
```

Will do the same as the previous example. But it will plot the first array element of a plane along column 3 against sequence number.

```
PLOT/TAB 3-Dtable #3 #5 ?  3:6:1,10 1,4
```

Plot the values found in a plane along column 3 against column 5. The data will be presented according to the following scheme:
array element: 1 2 3 4 5 6 7 8 etc.
line type : 1 4 1 4 1 4 1 4 ...
symbol type : 3 4 5 6 10 3 4 5 ...

```
PLOT/TAB 3-Dtable @3:name @5 ?  3:6:1,10 1,4
```

Take a vector at row 3, column "name" as abscissa and the plane along row 5 in the depth direction as ordinate. Plotted with the same line and symbol types as in the previous example. The vector and plane are both read (by default) array by array, thus there are no dimensional problems.

```
PLOT/TAB 3-Dtable [3]#1..4 #2[1..4] ?  3:6:1,10 1,4
```

Take plane along depth 3, from column 1 to 4 as abscissa and the plane along column 2, from depth 1 to 4 as ordinate. Both planes are read column by column so four lines will be plotted with resp. line type 1, 4, 1, 4.

# PLOT/VECTOR

## PLOT/VECTOR *core* 26-AUG-1991 RHW

**Purpose:** Plot vector map from two 2-dim. images with smoothing option

**Syntax:** `PLOT/VECTOR frame_a frame_b [coord_str] [x_sc,y_sc[,x_off,y_off]]`

`[scale_r] [range] [sm_par] [head]`

`frame_a` input frame containing the intensities

`frame_b` input frame containing the position angles

`coord_str` area to be plotted in standard MIDAS notation; see MIDAS Users Guide Chapter 3. Default is either any manual setting done with by the SET/GRAPHICS command, or the whole area.

`x_sc,y_sc,x_off,y_off` scale in the x- and y-direction, offset in the x- and y-direction. For the scaling x_sc and y_sc, positive numbers are interpreted as scale parameters (world units/mm), negative numbers as size parameters (axis will be made sc_* mm. long). A combination of a positive and negative number is allowed. By default the plot fills the device area.
x_off and y_off determine the distance of the lower left corner of the plot to the lower left corner of the graphic device, measured in mm. By default, the plot is put at the top left of the graphic device, allowing space for the various labels.

`scale_r` scale of the vector length in intensity units/mm. By default the command computes the average of the minimimum and maximum cut values in the descriptor of frame_a; this average intensity will correspond to a vector length of 10 mm.

`range` intensity and position angle (in degrees) ranges to be plotted. Default is all vectors (i.e. 0,0,0,360 = all intensities and all position angles). The position angle is defined using the standard astronomical convention: 0 is pointing to the top, and the angle increases counter-clockwise. For the length of the vector the absolute values of the pixels in the intensity image is taken; The postion angles are assumed positive.

`sm_par` smoothing parameter i.e. the number of pixels "around" the central pixel which define the neighbourhood of that pixel.
Total number of pixels is then: (2*sm_par + 1)**2 Only integer numbers are allowed. Default is 0

`head` integer number. If the vector should have an arrow at its end this parameter should be 1, else give a 0. Default is 1, i.e. the arrow is drawn.

**Note:** The coordinates for the area are optional. If one specifies a "?", the area will be take for the manual frame settings for the x- and y-axis. If "C" is given, the window is selected interactively on the display screen using the cursor facilities. The cursor will be set on automatically, if it is not already. To pick up the coordinates, switch the cursors ON and press ENTER. To exit, switch the cursors OFF and press ENTER.

**See also:** OVERPLOT/VECTOR, SHOW/GRAPHICS, SET/GRAPHICS, PLOT/AXES, PLOT/CONTOUR, OVERPLOT/CONTOUR, PLOT/GRAY, OVERPLOT/GRAY

**Examples:** `PLOT/VECT spir_i spir_p [@281,@281:@320,@320] ? 0.5 ? 1`
Plot the polarisation vector using the intensities from spir_i and the position angle information form spir_p. Use the default scale for the axes, and a scale of 0.5 units/mm for the size of the vectors (i.e. an intensity of 10 pixel units will make the vector 20 mm). Apply a box smoothing over 3x3 pixels.

## PRINT/ACAT *core* 03-JUN-1991 KB

**Purpose:** Print ASCII file catalog entries.

**Syntax:** `PRINT/ACAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of catalog; defaulted to active ASCII file catalog |
| `lowno,hino` | first and last entry to be listed; defaulted to 1,9999 |

**See also:** ASSIGN/PRINT, READ/ACAT

**Note:** Even if you use SET/ACAT to make an ASCII file catalog active, new files are not automatically added to that catalog.

**Examples:** `PRINT/ACAT ceresas 10,22`
Print all entries with entry number in [10,22] of ASCII catalog ceresas.cat

## PRINT/DESCRIPTOR *core* 31-MAY-1991 KB

**Purpose:** Print descriptor values.

**Syntax:** `PRINT/DESCR frame [descr_list] [disp_flag]`

| | |
|---|---|
| `frame` | name of data frame |
| `descr_list` | single descriptor name or several descriptors separated by a comma; defaulted to '*', i.e. all descriptors |
| `disp_flag` | B(rief) for short, F(ull) for extensive display of descr.; defaulted to B |

**See also:** ASSIGN/PRINT, READ/DESCR, SHOW/DESCR, WRITE/DESCR, DELETE/DESCR

**Note:** PRINT/DESCR produces the same output as READ/DESCR on the device specified via ASSIGN/PRINT.

**Examples:** `PRINT/DESCR matamoros histogram`
Print the contents of descriptor histogram of frame matamoros.bdf.

## PRINT/FCAT *core* 01-JUNE-1989 KB

**Purpose:** Print fit file catalog entries.

**Syntax:** `PRINT/FCAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of catalog; defaulted to active fit file catalog |
| `lowno,hino` | first and last entry to be listed; defaulted to 1,9999 |

**See also:** ASSIGN/PRINT, READ/FCAT

**Note:** Use SET/FCAT to make a fit file catalog active.

**Examples:** `PRINT/FCAT peras 10,22`
Print all entries with entry number in [10,22] of fit file catalog peras.cat

# PRINT/HELP

## PRINT/HELP *core* 21-JUN-1991 KB

**Purpose:** Print help information.

**Syntax:** `PRINT/HELP [help_topic]`

| | |
|---|---|
| `help_topic` | command (a) or command/qualifier (b)<br>(a), all command/qualifier combinations of command are printed<br>(b), complete information about the specific command/qualifier is printed<br>if help_topic is omitted, a general help menu is printed;<br>this is the default |

**Note:** Commands and qualifiers may be truncated to the significant number of characters.
The text will be printed on the print device which is specified via the ASSIGN/PRINT command.

**See also:** HELP, HELP/..., ASSIGN/PRINT

**Examples:** `PRINT/HELP LOAD`
Will print all different options (qualifiers) of command LOAD

`PRINT/HELP LOAD/IMAGE`
Will print complete help info about command LOAD/IMAGE

`PRINT/HELP BYE/`
Will print info about command BYE. Note the way we indicate a command/qualifier option, if no qualifier exists.

## PRINT/HISTOGRAM *core* 17-OCT-1983 JDP

**Purpose:** Print the statistics of a table column.

**Syntax:** `PRINT/HISTOGRAM table column [bin [min [max]]]`

| | |
|---|---|
| `table` | the table name |
| `column` | the column reference |
| `bin` | optional bin size |
| `min` | optional minimum value |
| `max` | optional maximum value |

**See also:** STATISTICS/TAB

**Note:** None.

**Examples:** `PRINT/HIST eso01 :RADVEL`
Print the statistics of column labelled RADVEL of table eso01.tbl .

PRINT/ICAT *core* 01-JUNE-1989 KB

**Purpose:** Print image catalog entries.

**Syntax:** `PRINT/ICAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of catalog; defaulted to active image catalog |
| `lowno,hino` | first and last entry to be listed; defaulted to 1,9999 |

**See also:** ASSIGN/PRINT, READ/ICAT

**Note:** Use SET/ICAT to make an image catalog active.

**Examples:** `PRINT/ICAT manzanas 10,22`
Print all entries with entry number in [10,22] of image catalog manzanas.cat

# PRINT/IMAGE

## PRINT/IMAGE *core* 6-MAR-1995 KB

**Purpose:** Print image data values on device specified via ASSIGN/PRINT.

**Syntax:** `PRINT/IMAGE frame_specs [pixel_specs] [hide_header_flag]`

`frame_specs` name of data frame
or CURSOR if cursor window is used to define the region of the displayed frame which will be printed;
or CURSOR,1 if only one cursor is used for printing single pixels of displayed frame;
if 'frame_specs' set to CURSOR the following parameters are ignored

`pixel_specs` xcoord,noval or xcoord,ycoord,noval or xcoord,ycoord,zcoord,noval,
depending on the number of dimensions of the frame;
with coords according to the MIDAS standard for coordinates:
a) coord = real no. => world coordinate
b) coord = @integer => pixel coordinate
c) coord = "<" => first pixel coord. (same as @1)
d) coord = ">" => last pixel coordinate;
or = [xstart,ystart:xend,yend]
or = tbname,T
to use a table 'tbname' which must contain the columns :XSTART, :YSTART, :XEND and :YEND defining the windows in the frame which are to be printed, only world coordinates are possible for the window specifications;
defaulted to print the first 20 values of specified frame; i.e. like "<,<,20" for a 2-dim frame

`hide_header_flag` to H, the frame name as well as the lines indicating line and pixel no. are suppressed, else not;
defaulted to N

**See also:** ASSIGN/PRINT, READ/IMAGE, WRITE/IMAGE, STATISTICS/IMAGE, GET/CURSOR

**Note:** PRINT/IMAGE produces the same output as READ/IMAGE on the device specified via ASSIGN/PRINT.
To use the cursor option you have to enter the complete string 'CURSOR', no abbreviation is possible.
The 'hide_header_flag' is especially useful for producing plain ASCII files, containing only the data values, as output.

**Examples:** `PRINT/IMAGE moctezuma 10.0,120.0,24`
Print 24 pixel values of image 'moctezuma.bdf' starting at x-world coord 10.0 and y-world coord 120.0

`PRINT/IMAGE moctezuma <,@9,10`
Print 10 pixel values of image 'moctezuma.bdf' starting at first pixel in the 9th line.

`PRINT/IMAGE moctezuma [<,@9:@10,@9]`
Same as above (assuming, that a line contains at least 10 pixels).

`PRINT/IMAGE CURSOR`
Move cursor window to define region on displayed frame and press ENTER on cursor board (or "enter button" of the mouse) to print all pixel values inside cursor window. To terminate, turn both cursors off and press ENTER (for DeAnza) or press exit button of the mouse (for XWindows)

`PRINT/IMAGE caballo asno,t`
Print pixels of frame 'caballo.bdf', the regions are defined in the columns :XSTART, :YSTART, :XEND, :YEND of table 'asno.tbl'.

## PRINT/KEYWORD *core* 13-FEB-1992 KB

**Purpose:** Print contents of keywords.

**Syntax:** `PRINT/KEYWORD [key_list] [since]`

`key_list` one or more keywords separated by a comma (no spaces!)
or '*' (wildcard option) where all keywords are printed
or a 'keyword catalog' indicated via extension ".cat";

defaulted to '*'

`since` lower update time limit;
only those keywords of 'key_list' which have been updated since that time are printed

**See also:** ASSIGN/PRINT, READ/KEYWORD, SHOW/KEYWORD, HELP/KEYWORD
also chapter 3 of the MIDAS Users Guide, volume A

**Note:** PRINT/KEY produces the same output as READ/KEY on the device specified via ASSIGN/PRINT.

Each keyword gets a timestamp when is is modified. This timestamp is a number giving the current time expressed in seconds elapsed since 1st January, 1970.
The parameter 'since' has to be given as such a number. To obtain the current time in that format use the M$SECS function in MIDAS.
A 'keyword catalog' is simply an ASCII file with records containing one or more keywords (separated by comma).

**Examples:** `PRINT/KEY in_a`
Print contents of keyword IN_A.

`PRINT/KEY keywo.cat`
Print contents of all keywords in the 'keyword catalog' keywo.cat.
E.g. if the ASCII file keywo.cat contains the two records:
in_a,in_b
inputi
contents of keywords in_a, in_b and inputi would be printed..

## PRINT/LOGFILE *core* 08-NOV-1988 KB

**Purpose:** Print the specified pages of the logfile.

**Syntax:** `PRINT/LOGFILE [page_specs]`

`page_specs` pa,pe or ps or -pl
print pages pa, ..., pe or print only page ps
or print last pl pages, defaulted to 1,9999

**See also:** LOG/ON, LOG/OFF, ASSIGN/PRINT

**Note:** All terminal I/O in MIDAS is saved in the MIDAS logfile (in ASCII) which is stored in MID_WORK.

**Examples:** `PRINT/LOG -1`
print last page of logfile

# PRINT/TABLE

## PRINT/TABLE *core* 17-OCT-1983 JDP

**Purpose:** Print table values on the device/file specified via ASSIGN/PRINT.

**Syntax:** `PRINT/TABLE table [column ...] [elem1 [elem2]] [N] [width]`

or

`PRINT/TABLE table [elem1 [elem2]] [form] [N]`

| | |
|---|---|
| `table` | the table name |
| `column` | column reference(s) of the listed column(s). All the columns are listed by default |
| `elem1` | sequence number of the element to be printed, or lower limit if elem2 is specified |
| `elem2` | upper sequence number to be printed |
| `form` | optional format file to control the output layout of the table |
| `N` | optional parameter; when used , the command will NOT print the header of the table |
| `width` | optional number of characters per line in the output file/device. This parameter has only an effect when no format file is specified and is defaulted to 80. When a format file is specified, the maximum number of characters per line will be calculated automatically. |

**See also:** READ/TABLE, ASSIGN/PRINT

**Note:** The output will be directed to the device given in the ASSIGN/PRINT command (default is the line printer).
If a format file is specified, the command will list ONLY the columns for which a DEFINE/FIELD statement is defined.
See the command CREATE/TAB for the description of the format files.

**Examples:** `PRINT/TABLE eso02 @1 @20`
This would print elements 1 to 20 of all columns in the table eso02.tbl .

`PRINT/TABLE mytable myform N`
This will print the table mytable.tbl without any header using the format file myform.fmt

## PRINT/TCAT *core* 01-JUNE-1989 KB

**Purpose:** Print table catalog entries.

**Syntax:** `PRINT/TCAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of catalog; defaulted to active table catalog |
| `lowno,hino` | first and last entry to be listed; defaulted to 1,9999 |

**See also:** ASSIGN/PRINT, READ/TCAT

**Note:** Use SET/TCAT to make a table catalog active.

**Examples:** `PRINT/TCAT duraznos 10,22`
Print all entries with entry number in [10,22] of table catalog duraznos.cat

## PROJECTION/TABLE *core* 22-MAY-1985 JDP + FO

**Purpose:** Project one or more columns of an input table into a new output table.

**Syntax:** `PROJECTION/TABLE intable outable column_selection`

| | |
|---|---|
| `intable` | name of input table |
| `outable` | name of (new) output table |
| `column_selection` | column(s) for which to realize the projection |

**Note:** none

**See also:** COPY/TT, COPY/TAB

**Examples:** `PROJECTION/TABLE MYTAB OUTAB #1`
The OUTAB table will contain only the first column from MYTAB

`PROJECTION/TABLE MYTAB OUTAB :X_COORD,Y_COORD`
OUTAB will contain the columns labelled X_COORD and Y_COORD from MYTAB

## READ/ACAT *core* 10-OCT-1988 KB

**Purpose:** Read entries of given ASCII Catalog.

**Syntax:** `READ/ACAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of ASCII Catalog; |
| `lowno,hino` | first entry no. and last entry no. to be listed; defaulted to 1,99999 |

**See also:** READ/ICAT, READ/TCAT, READ/FCAT
PRINT/ACAT, CREATE/ACAT, SHOW/ACAT, ADD/ACAT, SUBTRACT/ACAT

**Examples:** `READ/ACAT lolly 10,22`
Display all entries with entry number in [10,22] of ASCII catalog 'lolly.cat'.

# READ/COMMANDS

READ/COMMANDS *core* 03-JUL-1990 KB

**Purpose:** Read commands from a procedure + store them into command buffer, at the moment limited to 15 commands.

**Syntax:** `READ/COMMANDS [proc]`

| | |
|---|---|
| `proc` | name of the Midas procedure containing the commands we want to store into the command buffer;<br>defaulted to midtempZY.prg, where ZY is the current MIDAS unit, stored in MID_WORK |

**See also:** WRITE/COMMANDS, CLEAR/BUFFER

**Note:** We first look for the procedure in the current directory and then in MID_WORK.

**Examples:** `READ/COMMANDS mycomms`
Fill command buffer with commands stored in mycomms.prg .

READ/DESCR *core* 31-JAN-1994 KB

**Purpose:** Display descriptor values.

**Subject:** Descriptors, Data frames

**Syntax:** `READ/DESCR frame [descr_list] [disp_flag]`

| | |
|---|---|
| `frame` | name of data frame holding descriptor; default file type is .bdf; so if you want to see the descriptors of a table, you have to append also the type (e.g.: '.tbl') to the filename |
| `descr_list` | single descriptor name or several descriptors separated by a comma; defaulted to the standard descriptors |
| `disp_flag` | B(rief) or F(ull) or H(ide Header Info);<br>B: show only desriptor name(s) and value(s)<br>F: show also descriptor type and no. of elements;<br>H: show only desriptor value(s)<br>defaulted to B |

**See also:** WRITE/DESCR, SHOW/DESCR, PRINT/DESCR, DELETE/DESCR, COPY/DD

**Note:** If descr_list = '*', all descriptors of the frame are displayed.
A descriptor name may also be a pattern like 'abc*', then all descriptors beginning with 'abc' are displayed.
If descr_list is set to a specific descr. name, also the first 4 elements of integer keyword OUTPUTI are filled:
OUTPUTI(1) = 1 or 0, if specified descr. exists or not.
If OUTPUTI(1) is 1, then (2) = descr. type (1,2,3 or 4 for integer, real, character or double descr); (3) and (4) hold no. of elements and bytes per element of descr.
This information can then be used in a procedure.
For a description of the standard descriptors of an image or a table, see the MIDAS Environment document, Appendix C.

**Examples:** `READ/DESCR durazno STEP`
Display contents of descriptor STEP of frame 'durazno.bdf'.

`READ/DESCR durazno start,step full`
Display descriptors STEP and START of frame 'durazno.bdf' in extended format.

`READ/DESCR naranja.tbl *`
Display all descriptors of table file 'naranja.tbl'.

`READ/DESCR manzana wi*`
Display all descriptors of frame 'manzana.bdf' which begin with the string 'wi'.

# READ/FCAT

## READ/FCAT *core* 10-OCT-1988 KB

**Purpose:** Read entries of given FitFile Catalog.

**Syntax:** `READ/FCAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of FitFile Catalog;<br>defaulted to the currently active FitFile Catalog |
| `lowno,hino` | first entry no. and last entry no. to be listed;<br>defaulted to 1,99999 |

**See also:** READ/ICAT, READ/TCAT
SET/FCAT, CREATE/FCAT, SHOW/FCAT, ADD/FCAT, SUBTRACT/FCAT

**Note:** FitFiles in MIDAS commands may be accessed either by their name or by their entry number in a FitFile Catalog.
For VMS, note, that different versions of a file keep the same entry number, so you will always access the highest version of that file.

**Examples:** `READ/FCAT LOLLY 10,22`
Display all entries with entry number in [10,22] of FitFile Catalog lolly.cat .

`READ/FCAT`
Display all entries in the currently active FitFile Catalog.

## READ/FILE *core* 23-SEP-1992 KB

**Purpose:** Read an ASCII file which was opened before via the OPEN/FILE command.

**Syntax:** `READ/FILE file_id cbuf_key [maxrd]`

| | |
|---|---|
| `file_id` | file id which was returned by a previous OPEN/FILE command |
| `cbuf_key` | name of char. keyword which is getting the contents of the file read operation |
| `maxrd` | max. no. of chars. to read; defaulted to 200 |

**See also:** OPEN/FILE, CLOSE/FILE, WRITE/FILE

**Note:** The actual no. of characters read from the file is stored as the 2nd element of the integer keyword specified in the related OPEN/FILE command. If the end-of-file is reached, -1 will be stored there.
The 'cbuf_key' keyword will be filled always from the beginning.

**Examples:** `READ/FILE 8 outputc 20`
Read ASCII file with file_id 8, char. keyword OUTPUTC will contain max. 20 chars. read from file. The 2nd element of the integer keyword specified in the related OPEN/FILE command will hold the actual no. of characters read; if at EOF, that no. will be -1.

`READ/FILE fctr(1) outputc 20`
As above but use file_id which was stored in keyword fctr(1) before.

## READ/HISTOGRAM *core* 17-OCT-1983 JDP

**Purpose:** Read histogram of a column in the table.

**Syntax:** `READ/HISTOGRAM table column [bin [min [max]]]`

| | |
|---|---|
| `table` | table name |
| `column` | column reference |
| `bin` | optional size of the bins |
| `min` | optional minimum value |
| `max` | optional maximum value |

**See also:** STATISTICS/TAB

**Note:** none

**Examples:** `READ/HIS mytable :RADVEL`

## READ/ICAT *core* 10-OCT-1988 KB

**Purpose:** Read entries of given Image Catalog.

**Syntax:** `READ/ICAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of Image Catalog;<br>defaulted to the currently active Image Catalog |
| `lowno,hino` | first entry no. and last entry no. to be listed;<br>defaulted to 1,99999 |

**See also:** READ/TCAT, READ/FCAT
PRINT/ICAT, CREATE/ICAT, SHOW/ICAT, ADD/ICAT, SUBTRACT/ICAT

**Note:** Images in MIDAS commands may be accessed either by their name or by their entry number in an Image Catalog.
For VMS, note, that different versions of a file keep the same entry number, so you will always access the highest version of that file.

**Examples:** `READ/ICAT lolly 10,22`
Display all entries with entry number in [10,22] of image catalog 'lolly.cat'.

`READ/ICAT`
Display all entries in the currently active image catalog.

# READ/IMAGE

## READ/IMAGE *core* 06-MAR-1995 KB

**Purpose:** Display image data values.

**Syntax:** `READ/IMAGE frame_specs [pixel_specs] [hide_header_flag]`

`frame_specs` name of data frame
or CURSOR if cursor window is used to define the region of the displayed frame which will be read;
or CURSOR,1 if only one cursor is used for reading single pixels of displayed frame;
if 'frame_specs' set to CURSOR the following parameters are ignored

`pixel_specs` xcoord,noval or xcoord,ycoord,noval or xcoord,ycoord,zcoord,noval,
depending on the number of dimensions of the frame;
with coords according to the MIDAS standard for coordinates:
a) coord = real no. => world coordinate
b) coord = @integer => pixel coordinate
c) coord = "<" => first pixel coord. (same as @1)
d) coord = ">" => last pixel coordinate;
or = [xstart,ystart:xend,yend]
or = tbname,T
to use a table 'tbname' which must contain the columns :XSTART, :YSTART, :XEND and :YEND defining the windows in the frame which are to be read, only world coordinates are possible for the window specifications;
defaulted to read the first 20 values of specified frame; i.e. like "<,<,20" for a 2-dim frame

`hide_header_flag` to H, the frame name as well as the lines indicating line and pixel no. are suppressed, else not;
defaulted to N

**See also:** PRINT/IMAGE, GET/CURSOR, STATISTICS/IMAGE, WRITE/IMAGE

**Note:** To use the cursor option you have to enter the complete string 'CURSOR', no abbreviation is possible.
The 'hide_header_flag' is especially useful for producing plain ASCII files, containing only the data values, as output.

**Examples:** `READ/IMAGE chango @200,<,20`
Read (=display on terminal) 20 pixels of frame 'chango.bdf', starting at pixel no. 200 in x and first pixel in y.

`READ/IMAGE chango [@200,<:@219,<]`
Same as above (assuming, that a line contains at least 219 pixels).

`READ/IMAGE elefante 10.0,120.0,24 h`
Read 24 pixel values of image 'elefante.bdf' starting at x-world coord 10.0 and y-world coord 120.0; do not display frame name and line, pixel numbers.

`READ/IMAGE CURSOR`
Move cursor window to define region on displayed frame and press ENTER on cursor board (or "enter button" of the mouse) to display all pixel values inside cursor window. To terminate, turn both cursors off and press ENTER (for DeAnza) or press Exit (= right) button of the mouse (for XWindows)

`READ/IMAGE leon tigre,t`
Read pixels of frame 'leon.bdf', the regions are defined in the columns :XSTART, :YSTART, :XEND, :YEND of table 'tigre.tbl'.

READ/KEYWORD *core* 28-MAR-1994 KB

**Purpose:** Display contents of keywords.

**Syntax:** `READ/KEYWORD [key_list] [disp_flag] [since] [Midunit]`

| | |
|---|---|
| `key_list` | one or more keywords separated by a comma (no spaces!)<br>or 'xyz*' or '*xyz' where all matching keywords are displayed<br>or a 'keyword catalog' indicated via extension ".cat";<br>defaulted to '*' |
| `disp_flag` | '?', if you want the header line with the name, type and size of the keyword displayed as well,<br>= 'h', if you do not want this header line<br>defaulted to '?' |
| `since` | lower update time limit;<br>only those keywords of 'key_list' which have been updated since that time are displayed |
| `Midunit` | optional unit of another Midas session, then the keys are read from the keyfile of that session, this session must have the same Midas startup directory, i.e. same MID_WORK |

**See also:** PRINT/KEYWORD, WRITE/KEYWORD, COMPUTE/KEYWORD, COPY/KEYWORD, COPY/DK, COPY/KD, SHOW/KEYWORD, HELP/KEYWORD, DELETE/KEYWORD
also chapter 3 of the MIDAS Users Guide, volume A

**Note:** Each keyword gets a timestamp when is is modified. This timestamp is a number giving the current time expressed in seconds elapsed since 1st January, 1970.
The parameter 'since' has to be given as such a number. To obtain the current time in that format use the M$SECS function in MIDAS.
A 'keyword catalog' is simply an ASCII file with records containing one or more keywords (separated by a comma).

**Examples:** `READ/KEY IN_A`
Display contents of keyword IN_A.

`READ/KEY in_a h`
As above but omit header line.

`READ/KEY keywo.cat`
Display contents of all keywords with entries in the keyword catalog 'keywo.cat'.
E.g. if the ASCII file 'keywo.cat' contains the two records:
in_a,in_b
inputi
contents of keywords in_a, in_b and inputi would be displayed.

`READ/KEY in*,out* ? ? zx`
Display contents of all keywords beginning with 'IN' or 'OUT' as they are in the Midas session with unit ZX.

# READ/SETUP

## READ/SETUP *core* 27-JAN-1994 KB

**Purpose:** Read the contents of the variables related to a Setup.

**Syntax:** `READ/SETUP setup`

| | |
|---|---|
| `setup` | name of a Setup |

**See also:** INITIALIZE/SETUP, INFO/SETUP, WRITE/SETUP

**Note:** Setups are a collection of variables (keywords) which have to be set up before certain, more complex commands can be executed.
To get a list of all currently implemented Setups use the command INFO/SETUP.

**Examples:** `READ/SETUP catalog`
Display the contents of all the variables related to the Setup 'catalog'. These variables will be used by the EXECUTE/CATALOG command.

## READ/TABLE *core* 17-OCT-1983 JDP

**Purpose:** Read table elements.

**Syntax:** `READ/TABLE table [column_sel] [row_sel] [form]`

| | |
|---|---|
| `table` | table name |
| `column_sel` | column reference(s) of the listed column(s); the comma is used to separate columns. All the columns are listed by default. |
| `row_sel` | sequence numbers of the row(s) to be printed, commas can be used for enumeration, and a double dot for ranges All the rows are listed by default. |
| `form` | optional format file (type .FMT) |

**See also:** PRINT/TABLE, WRITE/TABLE, CREATE/TABLE

**Note:** If the format file is included, only the columns referenced by the DEFINE/FIELD command are listed. See the command CREATE/TABLE for description of the format files. The NULL values will be listed as a "*" for all types of columns except for character columns. In that case they will be listed as an empty field.

**Examples:** `READ/TABLE mytable :RA,DEC @1..10,1000,9999..`
Display the columns :RA and :DEC in rows 1 to 10, 1000, and from 9999 to the end of table 'mytable.tbl'.

READ/TCAT *core* 10-OCT-1988 KB

**Purpose:** Read entries of given Table Catalog.

**Syntax:** `READ/TCAT [cat_name] [lowno,hino]`

| | |
|---|---|
| `cat_name` | name of Table Catalog;<br>defaulted to the currently active Table Catalog |
| `lowno,hino` | first entry no. and last entry no. to be listed;<br>defaulted to 1,99999 |

**See also:** READ/ICAT, READ/FCAT
PRINT/TCAT, CREATE/TCAT, SHOW/TCAT, ADD/TCAT, SUBTRACT/TCAT

**Note:** Tables in MIDAS commands may be accessed either by their name or by their entry number in a Table Catalog.
For VMS, note, that different versions of a file keep the same entry number, so you will always access the highest version of that file.

**Examples:** `READ/TCAT lolly 10,22`
Display all entries with entry number in [10,22] of table catalog 'lolly.cat'.

`READ/TCAT`
Display all entries in the currently active table catalog.

# REBIN/II

## REBIN/II *core* 29-OCT-1985 MR

**Purpose:** Nonlinear rebinning of 1D data. Image to Image transformation.

**Syntax:** `REBIN/II outima inima refima [func] [param] [intop]`

| | |
|---|---|
| `outima` | name of output image |
| `inima` | name of input image |
| `refima` | reference image |
| `func` | function code (see below) for the mapping of OUTPUT indep.variable into INPUT indep. variable defaulted to LIN |
| `param` | parameters for the function used, defaulted to 0.,1. |
| `intop` | interpolation option (se below) defaulted to PIX |

**Note:**

FUNCTIONS (parameters A, B, C, ..... )
LIN x(old) = A + Bx
POL x = A + Bx +Cxx + .... (up to 16 coeff)
INV x = A + B/x
EXP x = A + B*EXP(Cx)
DEX x = A + B*10.**(Cx)
LOG x = A + B*LOG(Cx)
DLG x = A + B*LOG10(x)
IPO x = A + B/(x-D) + C/(x-D)**2 + ...
U01 x = user defined function of x.
INTOP (interpolation options)
PIX take fractions of pixel areas
LIN interpolate linearly and add polygons
SPG interpolate with hermite polynomials and integrate with Gaussian formula

Recommended option is SPG. LIN is faster and may provide sufficient accuracy whenever well behaved data are rebinned to larger stepsizes. PIX (same speed as LIN) will give adequate results only when rebinning from an oversampled array (very smal bins) into large bins

Defaults: REBIN/II OUT IN REF LIN 0.,1. PIX

**Examples:** `none`

REBIN/IT *core* 29-OCT-1985 MR

**Purpose:** Nonlinear rebinning of 1D data. Image to Table transformation.

**Syntax:** `REBIN/IT outtab i,d[,b] inima [func] [param] [intop]`

| | |
|---|---|
| `outtab` | name of output table |
| `inima` | name of input image |
| `i` | column reference for independent variable |
| `d` | column reference for dependent variable |
| `b` | [optional] column reference for binwidth |
| `func` | function code (see below) for the mapping of OUTPUT indep.variable into INPUT indep.variable defaulted to LIN |
| `param` | parameters for the function used, defaulted to 0.,1. |
| `intop` | interpolation option (se below), defaulted to PIX |

**Note:** FUNCTIONS (parameters A, B, C, ..... )
LIN x(old) = A + Bx
POL x = A + Bx +Cxx + .... (up to 16 coeff)
INV x = A + B/x
EXP x = A + B*EXP(Cx)
DEX x = A + B*10.**(Cx)
LOG x = A + B*LOG(Cx)
DLG x = A + B*LOG10(x)
IPO x = A + B/(x-D) + C/(x-D)**2 + ...
U01 x = user defined function of x.

INTOP (interpolation options)

PIX take fractions of pixel areas LIN interpolate linearly and add polygons SPG interpolate with hermite polynomials and integrate with Gaussian formula

Recommended option is SPG. LIN is faster and may provide sufficient accuracy whenever well behaved data are rebinned to larger stepsizes. PIX (same speed as LIN) will give adequate results only when rebinning from an oversampled array (very smal bins) into large bins

Defaults: REBIN/IT OUT #1,#2,#3 IN LIN 0.,1. PIX

**Examples:** `none`

# REBIN/LINEAR

## REBIN/LINEAR *core* 01-OCT-1992 KB

**Purpose:** Do linear rebinning of an image using new stepsizes and start points.

**Syntax:** `REBIN/LINEAR in out [stepx,stepy] [offx,offy] [startx,starty]`

`[fluxcons]`

`REBIN/LINEAR in out [refframe] [fluxcons]`

| | |
|---|---|
| `in` | input frame |
| `out` | output frame |
| `refframe` | optional reference frame; if given, the start (and with that the offsets 'offx,offy' will be determined) and step values for the output frame will be copied from the refframe |
| `stepx,stepy` | new stepsizes in x,y (in world coordinates);<br>defaulted to 1.,1. |
| `offx,offy` | absolute offset (in world coordinates) from start in x,y of input frame where we begin sampling for the result frame;<br>defaulted to 0.,0. |
| `startx,starty` | new start values (in world coords), if given, these values will be stored in descriptor START of result frame (but they do not affect the beginning of the sampling, that is controlled by 'offx,offy');<br>otherwise, descriptor START of the input frame + the offsets described above will determine the new values of START |
| `fluxcons` | YES or NO, only applicable for new larger stepsizes;<br>if YES the total flux should be conserved, i.e. if n pixels of value z are rebinned into one, this resulting pixel will have value n*z;<br>if NO, the resulting pixel will have value z;<br>so with NO you can do overlapping PLOT and OVERPLOT commands;<br>defaulted to NO |

**See also:** REBIN/SPLINE, REBIN/ROTATE, REBIN/II, REBIN/IT, REBIN/TI, REBIN/TT
READ/DESCR, WRITE/DESCR

**Note:** In MIDAS the following convention is used:
World coordinate values (e.g. start values) are related to the center of the pixels!
So, if you have e.g. start values 12.4, 4.8 and stepsize 1.4, 1.4 in a frame, the left and right edges of the first pixel (seen as a small plane) would be at 11.7 and 13.1 and the lower and upper edge at 4.1 and 5.5, respectively. The center of the first sampled pixel would be at 12.4+'offx',4.8+'offy'.
Pixels in the result frame sampled outside the input frame are set to the null value stored in real keyword NULL(2); the no. of pixels set to null will be stored in NULL(1). You can start sampling outside the input frame by specifying the offsets accordingly.

**Examples:** `REBIN/LINEAR conejo tortuga 0.3,.9`
Let us assume that the frame 'conejo.bdf' has start = 0.0,0.0 and step = 1.0,1.0.
Then we rebin it linearly with stepsizes 0.3 in x- and 0.9 in y-direction, we start sampling with the first pixel in x and y of 'conejo.bdf', keep same start values and store results in frame 'tortuga.bdf', the result frame has more pixels than 'conejo.bdf'.

`REBIN/LINEAR conejo tortuga 0.3,0.9 1.0,3.3`
As above but start sampling at pixel with has center 1.0 and 3.3, therefore 'tortuga.bdf' will have less pixels than in the ex. above. Descriptor START of tortuga will be 1.0,3.3 (0.0+1.0,0.0+3.3).

`REBIN/LINEAR conejo tortuga 0.3,0.9 1.,3.3 401.3,-20.8`
As above but descr. START of 'tortuga.bdf' will contain 401.3,-20.8 ; the data will be exactly the same.

`REBIN/LINEAR conejo pajaro mariposa`
Let us assume that the frame 'mariposa.bdf' has start = 1.2,1.8 and step = 1.6,1.6.
Then we rebin 'conejo.bdf' linearly with stepsizes 1.6 in x- and y-direction.
We start sampling with the pixel the center of which is located at 1.2 and 1.8. Keep intensity of new pixels at same scale and store results in frame 'pajaro.bdf'.

`REBIN/LINEAR conejo pajaro mariposa NO`
As above but new pixels will have approx. 1.6 times the intensity of the input pixels.

# REBIN/ROTATE

## REBIN/ROTATE *core* 04-MAY-1994 KB

**Purpose:** Rotate a 2-dim image by given angle around rotation point and rebin with new stepsizes.

**Syntax:** `REBIN/ROTATE in out [rot_specs] [ref_frame] [ref_flag]`

| | |
|---|---|
| `in` | name of input frame |
| `out` | name of output frame (will be created by this command) |
| `rot_specs` | angle,rotx,roty,scalx,scaly (a) or KEYWORD (b)<br>angle = rotation angle in degrees (positive angle leads to counterclockwise rotation);<br>rotx,roty = coordinates of point around which the image will be rotated, or C for using coordinates of center pixel as "rotation point";<br>scalx,scaly = scaling factors in x and y, the stepsizes of the output frame will be: stepsize of input frame * scaling factor;<br>if no reference frame is given (see following par.), the start of the result frame will be set so, that the coordinates of this "rotation point" will remain the same in the resulting image;<br>if rot-specs = string KEYWORD then the transformation matrix for the rotation is taken from the keyword TRANSFRM which should have been filled by the Midas command ALIGN/IMAGE before;<br>defaulted to 45.,C,C,1.,1. (i.e. option (a)) |
| `ref_frame` | name of optional 2-dim reference frame;<br>if a reference frame is given, the start and stepsize for the output frame will be aligned with the reference frame (that means that scalx,scaly of parameter 'rot_specs' will be overruled by the stepsize of the reference frame);<br>This is also valid, if option (b) is used. |
| `ref_flag` | YES or NO - if set to YES the output frame will have the exact start values, step size and no. of pixels be as in the reference frame (only applicable if reference frame given and useful for the KEYWORD option (b));<br>defaulted to NO |

**See also:** ROTATE/CLOCK, ROTATE/COUNTERCLOCK, ALIGN/IMAGE, REBIN/LINEAR

**Note:** If no reference frame is given, the output frame will be the smallest rectangle containing the rotated image, otherwise it has the size of the reference frame.
All pixels in the output frame which are outside the rotated image are set to a user defined Nullvalue (2. element of keyword NULL).
The input image is first rotated and then the pixels of the result frame are sampled with the stepsize defined by saclx,scaly
If ref_flag is not set to YES, the result frame might look as if it were not aligned with the reference frame (especially, when there was only a very small rotation angle). But if you e.g. divide these two frames, you see that the objects you used for the ALIGN/IMAGE command are aligned correctly.

**Examples:** `REBIN/ROTA conejo tortuga 23.4,22.,31.`
Rotate image 'conejo.bdf' by 23.4 degrees around point with coords. (22.0,31.0) - this is not necessarily a pixel of the image;
result will be stored in frame 'tortuga.bdf'

`REBIN/ROTA conejo tortuga 23.4,C,31.,1.5,1.5`
Rotate around point with x-coord. = coord. of x-center pixel and y-coord. = 31.0. Rebin output image 'tortuga.bdf' with stepsize which is 1.5 * stepsize of 'conejo.bdf'.

`REBIN/ROTA conejo tortuga 23.4,C,31.,1.5,1.5 ballena`
As above, but the size of the output frame is determined by the size of the reference frame 'ballena.bdf' and the scaling values are taken as the ratio of the stepsize of 'ballena.bdf' and 'conejo.bdf' (i.e. the scale values 1.5,1.5 are overriden!).

`REBIN/ROTA conejo tortuga KEYWORD ballena yes`
Rotate image 'conejo.bdf' and use keyword TRANSFRM (which was setup before via the ALIGN/IMA command) to get rotation angle, sample offsets and scaling factors.
Image 'tortuga.bdf' will have the start coords., step sizes and no. of pixels as image 'ballena.bdf'.

# REBIN/SPLINE

## REBIN/SPLINE *core* 09-JAN-1992 KB

**Purpose:** Do rebinning of an image via cubic splines using new stepsizes and start points.

**Syntax:** `REBIN/SPLINE in out [stepx,stepy] [offx,offy] [startx,starty]`

`REBIN/SPLINE in out [refframe]`

| | |
|---|---|
| `in` | input frame |
| `out` | output frame |
| `refframe` | optional reference frame; if given, the start (and with that the offsets 'offx,offy' will be determined) and step values for the output frame will be copied from the refframe |
| `stepx,stepy` | new stepsizes in x,y (in world coordinates); defaulted to 1.,1. |
| `offx,offy` | absolute offset (in world coordinates) from start in x,y of input frame where we begin sampling for the result frame; defaulted to 0.,0. |
| `startx,starty` | new start values (in world coords), if given, these values will be stored in descriptor START of result frame (but they do not affect the beginning of the sampling, that is controlled by 'offx,offy'); otherwise, descriptor START of the input frame + the offsets described above will determine the new values of START |

**Note:** In MIDAS the following convention is used:
World coordinate values (e.g. start values) are related to the center of the pixels!
So, if you have e.g. start values 12.4, 4.8 and stepsize 1.4, 1.4 in a frame, the left and right edges of the first pixel (seen as a small plane) would be at 11.7 and 13.1 and the lower and upper edge at 4.1 and 5.5, respectively. The center of the first sampled pixel would be at 12.4+'offx',4.8+'offy'.
Pixels in the result frame sampled outside the input frame are set to the null value stored in real keyword NULL(2); the no. of pixels set to null will be stored in NULL(1). You can start sampling outside the input frame by specifying the offsets accordingly.

**See also:** REBIN/LINEAR, REBIN/ROTATE, REBIN/II, REBIN/IT, REBIN/TI, REBIN/TT, READ/DESCR, WRITE/DESCR

**Examples:** `REBIN/SPLINE conejo tortuga 0.3,.9`
Let us assume that the frame 'conejo.bdf' has start = 0.0,0.0 and step = 1.0,1.0.
Then we rebin it with stepsizes 0.3 in x- and 0.9 in y-direction, we start sampling with the first pixel in x and y of 'conejo.bdf', keep same start values and store results in frame 'tortuga.bdf', the result frame has more pixels than 'conejo.bdf'.

`REBIN/SPLINE conejo tortuga 0.3,0.9 1.0,3.3`
As above but start sampling at pixel with has center 1.0 and 3.3, therefore 'tortuga.bdf' will have less pixels than in the ex. above. Descriptor START of tortuga will be 1.0,3.3 (0.0+1.0,0.0+3.3).

`REBIN/SPLINE conejo tortuga 0.3,0.9 1.,3.3 401.3,-20.8`
As above but descr. START of 'tortuga.bdf' will contain 401.3,-20.8 ; the data will be exactly the same.

`REBIN/SPLINE conejo pajaro mariposa`
Let us assume that the frame 'mariposa.bdf' has start = 1.2,1.8 and step = 0.6,0.6.
Then we rebin 'conejo.bdf' with stepsizes 0.6 in x- and y-direction.
We start sampling with the pixel the center of which is located at 1.2 and 1.8. Store results in frame 'pajaro.bdf'.

REBIN/TI *core* 29-OCT-1985 MR

**Purpose:** Nonlinear rebinning of 1D data. Table to Image transformation.

**Syntax:** `REBIN/TI outima intab i,d[,b] refima [func] [param] [intop]`

| | |
|---|---|
| `outima` | name of output image |
| `intab` | name of input table |
| `refima` | reference image |
| `i` | column reference for independent variable |
| `d` | column reference for dependent variable |
| `b` | [optional] column reference for binwidth |
| `func` | function code (see below) for the mapping of OUTPUT indep.variable to INPUT indep.variable defaulted to LIN |
| `param` | parameters for the function used, defaulted to 0.,1. |
| `intop` | interpolation option (se below), defaulted to PIX |

**See also:** CONVERT/TABLE, INTERPOLATE/TI

**Note:** FUNCTIONS (parameters A, B, C, ..... )
LIN x(old) = A + Bx
POL x = A + Bx +Cxx + .... (up to 16 coeff)
INV x = A + B/x
EXP x = A + B*EXP(Cx)
DEX x = A + B*10.**(Cx)
LOG x = A + B*LOG(Cx)
DLG x = A + B*LOG10(x)
IPO x = A + B/(x-D) + C/(x-D)**2 + ...
U01 x = user defined function of x.

INTOP (interpolation options)

PIX take fractions of pixel areas LIN interpolate linearly and add polygons SPG interpolate with hermite polynomials and integrate with Gaussian formula

Recommended option is SPG. LIN is faster and may provide sufficient accuracy whenever well behaved data are rebinned to larger stepsizes. PIX (same speed as LIN) will give adequate results only when rebinning from an oversampled array (very smal bins) into large bins

Defaults: REBIN/TI OUT IN #1,#2,#3 REF LIN 0.,1. PIX

**Examples:** `none`

# REBIN/TT

## REBIN/TT *core* 29-OCT-1985 MR

**Purpose:** Nonlinear rebinning of 1D data. Table to Table transformation.

**Syntax:** `REBIN/TT outtab i,d[,b] intab i,d[,b] [func] [parm] [intop]`

| | |
|---|---|
| `outtab` | name of output table |
| `intab` | name of input table |
| `i` | column reference for independent variable |
| `d` | column reference for dependent variable |
| `b` | [optional] column reference for binwidth |
| `func` | function code (see below) for the mapping of OUTPUT indep.variable to INPUT indep.variable defaulted to LIN |
| `param` | parameters for the function used, defaulted to 0.,1. |
| `intop` | interpolation option (se below), defaulted to PIX |

**Note:** FUNCTIONS (parameters A, B, C, ..... )
LIN x(old) = A + Bx
POL x = A + Bx +Cxx + .... (up to 16 coeff)
INV x = A + B/x
EXP x = A + B*EXP(Cx)
DEX x = A + B*10.**(Cx)
LOG x = A + B*LOG(Cx)
DLG x = A + B*LOG10(x)
IPO x = A + B/(x-D) + C/(x-D)**2 + ...
U01 x = user defined function of x.

INTOP (interpolation options)

PIX take fractions of pixel areas LIN interpolate linearly and add polygons SPG interpolate with hermite polynomials and integrate with Gaussian formula

Recommended option is SPG. LIN is faster and may provide sufficient accuracy whenever well behaved data are rebinned to larger stepsizes. PIX (same speed as LIN) will give adequate results only when rebinning from an oversampled array (very smal bins) into large bins

Defaults: REBIN/TT OUT #1,#2,#3 IN #1,#2,#3 LIN 0.,1. PIX

**Examples:** `none`

## REGRESSION/LINEAR *core* 17-OCT-1983 JDP

**Purpose:** Performs linear regression between table columns.

**Syntax:** `REGRESSION/method table y x1,x2,...`

| | |
|---|---|
| `table` | table name |
| `y` | column acting as dependent variable |
| `x1,x2,...` | columns acting as independent variables |

**Output:** Output from the regression is listed on terminal and stored in the following keywords:
OUTPUTC - character information
OUTPUTI(1)- N,no.of data,
(2)- M,no.of ind.var.
(3)- col.no. of dep.var.
(i)- col.no. of ind.var.
i=4,...,M+3
OUTPUTD(1)- Constant term
(i)- Coefficients
i=2,...,M+1
OUTPUTR(1)- F-value for the analysis of variance
(2)- Standard error of the estimate
(3)- Standard error of constant term
(i)- Standard error of coeffs.
i=4,...,M+3

**See also:** SAVE/REGRESSION, COMPUTE/REGRESSION, REGRESSION/POLYNOMIAL

**Note:** none

**Examples:** `REGRESSION/LINEAR data :Z :X,:Y`
Use the columns labeled :Z, :X, :Y of table 'data.tbl' and fit the function :Z = a*:X + b*:Y + c to determine the values of a, b and c.

`REGRESSION/LINEAR dito #2 #1`
Use the columns no. 2, no. 1 of table 'dito.tbl' and fit the function #2 = a*#1 + b to determine the values of a and b.

# REGRESSION/POLYNOMIAL

## REGRESSION/POLYNOMIAL *core* 28-FEB-1984 JDP

**Purpose:** Polynomial fit of one or two variables between table columns.

**Syntax:** `REGRESSION/POLYNOMIAL table y x1[,x2] degree1[,degree2]`

| | |
|---|---|
| `table` | the table name |
| `y` | column acting as dependent variable |
| `x1,x2` | column(s) acting as independent variable(s) |

**Output:** Output from the fit is listed on terminal and stored in the following keywords:
OUTPUTC - character information
OUTPUTR(1)- minimum value of x
(2)- maximum value of x
(3)- minimum value of y (optional)
(4)- maximum value of y (optional)
(5)- Standard error of the estimate
OUTPUTI(1)- N,no.of data,
(2)- M,no.of ind.var.
(3)- col.no. of dep.var. (4)- col.no. of indep.var.
(5)- degree (ND)
OUTPUTD(i)- Coefficients
i=1,...,ND+1

**Note:** none

**See also:** SAVE/REGRESSION, COMPUTE/REGRESSION

**Examples:**

```
REGRESSION/POLYNOMIAL DATA :Z :X 2

SAVE/REGR DATA COEF

COMPUTE/REGR DATA :FIT = COEF
```

will fit the function :Z = a0 + a1*:X + a2*:X**2, save the coefficients in the table DATA with the name COEF and compute the result of the linear fit in column :FIT.

```
REGRESSION/POLYNOMAIL DATA :Z :X,:Y 2,2

SAVE/REGR DATA COEF

COMPUTE/REGR DATA :FIT = COEF
```

will fit the function :Z = a0 + a1*:X + a2*:X**2 + a3*:Y+ a4*:X*:Y + a5*:X**2:Y + a6*:Y**2 + a7*:X*Y**2 + a8*:X**2*:Y**2 save the coefficients in the table DATA with the name COEF

## REGRESSION/TABLE *core* 28-OCT-1985 JDP

**Purpose:** Polynomial fit of one or two variables between two tables. The command can be used to find the correspondence between the entries in the two tables. There are two modes of operation:
1. A initial correspondance between the entries is defined by the REFERENCE columns in both tables. (see command SET/REFERENCE)
2. Initial guesses for the correspondence are given as a set of coeffs.
The reference column of the first input table is updated to include the references of the correspondent entries in the second input table.

**Syntax:** `REGRESSION/TABLE table1 x1[,x2] table2 y1[,y2] degree tol [guess]`

| | |
|---|---|
| `table1` | first table name |
| `x1,[x2]` | columns with the values to compare |
| `table2` | second table name |
| `y1,[y2]` | columns with the values to compare |
| `degree` | degree of the polynomial used in the transformation (1D or 2D) |
| `tol` | tolerance in the correspondance |
| `guess` | optional guesses for the transformation in the second mode |

**Output:** Output coefficients are listed on terminal and stored in the following keywords:
OUTPUTC - character information
OUTPUTR(5)- Standard error of the estimate
OUTPUTI(1)- N,no.of data,
(2)- M,no.of ind.var.
(4)- col.no. of indep.var.
(5)- degree (ND)
OUTPUTD(i)- Coefficients
i=1,...,ND+1

**Note:** 1. The main applications are
- In spectroscopy to compute the dispersion coefficients, by comparing the user table with line positions and initial identifications (table1) with the line catalogue (table2).
- In astrometry to compare object positions from to tables.
2. The coefficients can be saved (SAVE/REGRESSION) and used latter to compute fitted values (COMPUTE/REGRESSION).
3. The command requires to set the reference columns in both tables.

**Examples:** `An application of REGRESSION/TABLE`
The following sequence will identify entries in mytable and will compute the dispersion coefficients in a 1D application:

```
IDENTIFY/GCURSOR mytable :IDENT :X

SET/REFERENCE mytable :IDENT

SET/REFERENCE lincat :WAVE

REGRESSION/TABLE mytable :X LINCAT :WAVE 5 0.1

SAVE/REGRESSION mytable DISPCOEF
```

The command is not yet implemented

# RENAME/FIT

## RENAME/FIT *core* 04-NOV-1994 KB

**Purpose:** Rename a fit file.

**Syntax:** `RENAME/FIT old new [history] [overwrite]`

| | |
|---|---|
| `old` | old name of fit frame |
| `new` | new name of fit frame |
| `history` | flag for update of HISTORY descriptor;<br>if NO, the history descriptor of the new frame is not updated;<br>defaulted to YES |
| `overwrite` | flag for overwrite check;<br>if CONFIRM, we check, if a file with the new name already exists and if so, ask for confirmation before overwriting it;<br>defaulted to NO_CONFIRM |

**See also:** RENAME/IMAGE, RENAME/TABLE

**Note:** The parameters may also be referenced via:
OLD= NEW= HISTORY= OVERWRITE=
The file is renamed and if a fitfile catalog is enabled its entry in there is updated as well.

**Examples:** `RENAME/FIT ballena tiburon`
Rename the fit file ballena.fit to tiburon.fit, if tiburon.fit already exists it is overwritten (Unix) or a new version created (VMS). The descriptor HISTORY of tiburon.fit is updated.

`RENAME/FIT gato gata ? confirm`
Rename fit file gato.fit to gata.fit, if gata.fit already exists we ask for confirmation before actually overwriting that file. If so, descriptor HISTORY of gata.fit is updated.

RENAME/IMAGE *core* 04-NOV-1994 KB

**Purpose:** Rename an image frame.

**Syntax:** `RENAME/IMAGE old new [history] [overwrite]`

| | |
|---|---|
| `old` | old name of image frame |
| `new` | new name of image frame |
| `history` | flag for update of HISTORY descriptor;<br>if NO, the history descriptor of the new frame is not updated;<br>defaulted to YES |
| `overwrite` | flag for overwrite check;<br>if CONFIRM, we check, if a file with the new name already exists and if so, ask for confirmation before overwriting it;<br>defaulted to NO_CONFIRM |

**See also:** RENAME/FIT, RENAME/TABLE

**Note:** The parameters may also be referenced via:
OLD= NEW= HISTORY= OVERWRITE=
The frame is renamed and its entry in the active image catalog (if enabled) is updated as well.

**Examples:** `RENAME/IMAGE gallina gallo`
Rename file gallina.bdf to gallo.bdf, if gallo.bdf already exists it is overwritten (Unix) or a new version created (VMS). The descriptor HISTORY of gallo.bdf is updated.

`RENAME/IMAGE gato gata ? confirm`
Rename file gato.bdf to gata.bdf, if gata.bdf already exists we ask for confirmation before actually overwriting that file. If so, descriptor HISTORY of gata.bdf is updated.

# RENAME/TABLE

## RENAME/TABLE *core* 04-NOV-1994 KB

**Purpose:** Rename a table frame.

**Syntax:** `RENAME/TABLE old new [history] [overwrite]`

| | |
|---|---|
| `old` | old name of table frame |
| `new` | new name of table frame |
| `history` | flag for update of HISTORY descriptor;<br>if NO, the history descriptor of the new table is not updated;<br>defaulted to YES |
| `overwrite` | flag for overwrite check;<br>if CONFIRM, we check, if a file with the new name already exists and if so, ask for confirmation before overwriting it;<br>defaulted to NO_CONFIRM |

**See also:** RENAME/FIT, RENAME/IMAGE

**Note:** The parameters may also be referenced via:
OLD= NEW= HISTORY= OVERWRITE=
The frame is renamed and its entry in the active table catalog (if enabled) is updated as well.

**Examples:** `RENAME/TABLE cavallo asno`
Rename table file cavallo.tbl to asno.tbl, if asno.tbl already exists it is overwritten (Unix) or a new version created (VMS). The descriptor HISTORY of asno.tbl is updated.

`RENAME/TABLE gato gata ? confirm`
Rename table file gato.tbl to gata.tbl, if gata.tbl already exists we ask for confirmation before actually overwriting that file. If so, descriptor HISTORY of gata.tbl is updated.

REPLACE/IMAGE *core* 30-JAN-1990 KB

**Purpose:** For all pixels in a test frame with intensities in a given interval the values of the corresponding pixels of the input frame are replaced with the result of an algebraic expression.

**Syntax:** `REPLACE/IMAGE in out [test/]low,hi=express1[,express2]`

| | |
|---|---|
| `in` | primary input frame |
| `out` | output frame |
| `test` | optional different test frame, if omitted (also no '/'), the primary input frame will be used as test frame |
| `low,hi` | lower + upper limit (as real numbers) of intensity interval in the test frame for replacements;<br>low may be set to "<" to indicate neg. infinity<br>hi may be set to ">" to indicate pos. infinity |
| `express1` | algebraic expression to calculate the replacement value for all pixels in the interval [low,hi]<br>a) real constant<br>b) auxiliary input frame<br>c) aux op constant, op = basic operation (+, -, * or /)<br>The blanks above are just for better readability, NO blanks are possible in the actual expression (e.g. ima/12.3).<br>Auxiliary frame must come first, constant last. |
| `express2` | optional algebraic expression (same syntax as express1) to calculate the value for all pixels with intensities outside the interval [low,hi] |

**See also:** FILTER/IMAGE, COMPUTE/IMAGE, REPLACE/POLYGON
application procedure 'replace.prg'

**Note:** All images involved must have same dimensions!
Outside the interval [low,hi] the output frame will be equal to the input frame. unless a second expression is given.
To use more complicated expressions than the ones indicated above for the parameter 'express1' or 'express2', use the Midas command "@a replace" (Help for that is obtained via "HELP/APPLIC replace").

**Examples:** `REPLACE/IMAGE a c 2.001,2.02=2.1`

is equivalent to:
if ((a(n).ge.2.001).and.(a(n).le.2.02)) then
c(n) = 2.1
else
c(n) = a(n)
endif

`REPLACE/IMAGE a c 2.001,2.02=b`

is equivalent to:
if ((a(n).ge.2.001).and.(a(n).le.2.02)) then
c(n) = b(n)
else
c(n) = a(n)
endif

# REPLACE/IMAGE

```
REPLACE/IMAGE a c 2.001,2.02=b,-9.9
```

is equivalent to:
if ((a(n).ge.2.001).and.(a(n).le.2.02)) then
c(n) = b(n)
else
c(n) = -9.9
endif

```
REPLACE/IMAGE a c t/20.,22.=2.1
```

is equivalent to:
if ((t(n).ge.20.).and.(t(n).le.22.)) then
c(n) = 2.1
else
c(n) = a(n)
endif

```
REPLACE/IMAGE a c t/2.001,2.02=b+4.4,d
```

is equivalent to:
if ((t(n).ge.2.001).and.(t(n).le.2.02)) then
c(n) = b(n) + 4.4
else
c(n) = d(n)
endif

```
REPLACE/IMAGE a c t/<,2.02=b+4.4
```

is equivalent to:
if (t(n).le.2.02) then
c(n) = b(n) + 4.4
else
c(n) = a(n)
endif

## REPLACE/POLY *core* 07-JULY-1986 KB

**Purpose:** For all pixels in test frame inside a polygon the corresponding pixels of input frame are replaced with the value.

**Syntax:** `REPLACE/POLYGON in,intab out test/low,hi=value`

| | |
|---|---|
| `in` | primary input frame |
| `intab` | optional table with coordinates of the windows surrounding the polygons |
| `out` | output frame |
| `test` | optional different test frame;<br>if omitted (also no '/') the primary input frame will be used as test frame |
| `low,hi` | real low + high threshold for polygon detection in the test frame, pixels inside and on the polygon are replaced |
| `value` | replacement value, may be real constant (a) or auxiliary input frame (b) |

**See also:** REPLACE/IMAGE, FIND/PIXEL

**Note:** All images involved must have same dimensions!
Outside the polygon the output frame will be set equal to the input frame.

**Examples:** `REPLACE/POLYGON gustavo,adolfo apumanque 2.001,2.02=vitacura`
Use table 'adolfo.tbl' to define the subframes of image 'gustavo.bdf' we work on. Inside these windows look for polygons with boundaries of intensity in the interval [2.001,2.02].
Replace all pixels of 'gustavo.bdf' inside these polygons with the corresponding pixel values of the image 'vitacura.bdf' and store results in image 'apumanque.bdf'.

## REPORT/PROBLEM *core* 27-OCT-1992 KB

**Purpose:** Send error reports and comments to the person(s) in charge of MIDAS.

**Syntax:** `REPORT/PROBLEM [errfile]`

| | |
|---|---|
| `errfile` | optional name of text file containing the MIDAS report |

**See also:** HELP [News], CREATE/GUI Help

**Note:** If you do not enter the name of a text file with an already prepared report, you write your comments, suggestions or error report into a text file (named 'report.err') using a text editor.
This file, together with some additional information about the current MIDAS session, will then be e-mailed to the person in charge of MIDAS at your site.
The editor is specified via the MIDAS command SET/MIDAS_SYSTEM, by default 'vi' (Unix) or 'EDIT' (VMS) is used.
Please, use this command frequently to report errors you find in MIDAS commands, suggestions for improvements, typing errors in the documentation, etc.

**Examples:** `REPORT/PROBLEM`
Use a text editor to fill a file with your report. Add info about the current MIDAS session and e-mail it to the local MIDAS responsible.

`REPORT/PROBLEM comments.dat`
Take the text file 'comments.dat' as input, add info about the current MIDAS session and e-mail it to the local MIDAS responsible.

# RESET/DISPLAY

## RESET/DISPLAY *core* 20-SEP-1990 KB

**Purpose:** Reset Xwindow display after Control C.

**Syntax:** `RESET/DISPLAY`

**See also:** CREATE/DISPLAY, DELETE/DISPLAY

**Note:** This command may be also used if the IDI server aborts due to other reasons and your windows disappear from the screen!

**Examples:** `RESET/DISPLAY`

RESTORE/NAME *core* 04-NOV-1994 KB

**Purpose:** Change file name to the name stored in descr. FILENAME

**Syntax:** `RESTORE/NAME [file_spec] [verbose] [history] [overwrite] [descr]`

| | |
|---|---|
| `file_spec` | name of single file or catalog name (with type '.cat');<br>defaulted to currently active image catalog |
| `verbose` | YES or NO, to display new name for each file as well as error messages or comments;<br>defaulted to YES |
| `history` | flag for update of HISTORY descriptor;<br>if NO, the history descriptor of the new frame is not updated;<br>defaulted to NO |
| `overwrite` | flag for overwrite check;<br>if CONFIRM, we check, if a file with the new name already exists and if so, ask for confirmation before overwriting it;<br>defaulted to NO_CONFIRM |
| `descr` | name of character descriptor which should be used, if the descr. FILENAME does not exist for an input file;<br>defaulted to IDENT |

**See also:** INTAPE/FITS, INDISK/FITS, CREATE/ICAT, READ/DESCR, RENAME/IMAGE

**Note:** The parameters may also be referenced via:
FILE_SPEC= VERBOSE= HISTORY= OVERWRITE= DESCR=
Images, tables and fit files are supported as well as catalogs of these.
This command is especially useful after the INTAPE/FITS command in order to get the original filenames back.

**Examples:** `RESTORE/NAME`
Use the currently active image catalog. For each image frame in that catalog look for the char. descriptor FILENAME. If found, rename the frame to the content of FILENAME. Display new name for each image.

`RESTORE/NAME dosequis.bdf`
If the image 'dosequis.bdf' has a char. descriptor FILENAME containing e.g. the string 'cartablanca.bdf' it will be renamed to cartablanca.bdf .

`RESTORE/NAME tecate.cerveza no ? conf`
Rename file 'tecate.cerveza' to the content of descriptor FILENAME, e.g. 'paulaner.bier'. If a file named 'paulaner.bier' already exists the user is asked for confirmation before actually renaming the input file. The file type, e.g. table, will be determined and any active table catalog is updated. The new name of 'tecate.cerveza' is not displayed.

`RESTORE/NAME bohemia.cat descr=fitname`
Rename all files in catalog bohemia. The actual type of the catalog, e.g. fit file catalog, will be determined and all fit files with entries in 'bohemia.cat' will be renamed according to the content of descriptor FILENAME. If the descr. FILENAME does not exist for one of the input fit files, we try to use descr. FITNAME instead.

# RETRO/TABLE

## RETRO/TABLE *core* 07-DEC-1990 FO

**Purpose:** Convert a 3-D Table (created with 90NOV Version) to an Old Format

**Syntax:** `RETRO/TABLE table`

| | |
|---|---|
| `table` | table name |

**Note:** This conversion is not always possible, since the 3D table system offers many more possibilities. Format extensions (e.g. dates, sexagesimal) will be removed.

**Examples:** `RETRO/TABLE mytable`
Convert mytable into old format

## ROTATE/1DIM *core* 11-DEC-1986 KB

**Purpose:** Rotate a 1dim profile around its startpoint.

**Syntax:** `ROTATE/1DIM in out [nop_flag]`

| | |
|---|---|
| `in` | 1-dim input frame |
| `out` | 2-dim output frame |
| `nop_flag` | optional number-of-pixel flag;<br>ODD - no. of pixels in x- and y-direction of output frame will be: 2*input_nopix - 1<br>EVEN - no. of pixels will be: 2*input_nopix;<br>defaulted to ODD |

**See also:** REBIN/ROTATE, DECONVOLVE/IMAGE, CREATE/IMAGE, CREATE/FILTER GROW/IMAGE

**Note:** This command should be helpful if you need to create pointspread functions or filters. Just define the cross-section and then use this rotation.
Pixels outside the original domain of the 1-dim frame, i.e. the pixels in the corners of the 2-dim result frame, will be set to the value in the keyword NULL.

**Examples:** `ROTATE/1DIM profile psf`
Rotate 1-dim frame 'profile.bdf' and create 2-dim frame 'psf.bdf' with the ODD option.

## ROTATE/CLOCK *core* 11-DEC-1984 KB

**Purpose:** Rotate an image by multiples of 90 degrees clockwise.

**Syntax:** `ROTATE/CLOCK in out [factor]`

| | |
|---|---|
| `in` | input frame (2-dim image) |
| `out` | output frame (will be created by this command) |
| `factor` | 1, 2 or 3 for 90, 180 or 270 degrees rotation; defaulted to 1, i.e. 90 degrees |

**Note:** This command is not a rotation around a given point but strictly speaking just a rearrangement of the image pixels. Values for start and stepsize of result frame are adjusted accordingly.
Using REBIN/ROTATE with a rotation angle of -90.0 degrees will yield the same data in the result frame as ROTATE/CLOCK, but start and stepsize will be different!

**See also:** ROTATE/COUNTER_CLOCK, REBIN/ROTATE

**Examples:** `ROTATE/CLOCK paloma blanca`
rotate image paloma.bdf by 90 degrees clockwise and store result into frame blanca.bdf

## ROTATE/COUNTER_CLOCK *core* 11-DEC-1984 KB

**Purpose:** Rotate an image by multiples of 90 degrees counter_clockwise.

**Syntax:** `ROTATE/COUNTER_CLOCK in out [factor]`

| | |
|---|---|
| `in` | input frame (2-dim image) |
| `out` | output frame (will be created by this command) |
| `factor` | 1, 2 or 3 for 90, 180 or 270 degrees rotation; defaulted to 1, i.e. 90 degrees |

**Note:** This command is not a rotation around a given point but strictly speaking just a rearrangement of the image pixels. Values for start and stepsize of result frame are adjusted accordingly.
Using REBIN/ROTATE with a rotation angle of 90.0 degrees will yield the same data in the result frame as ROTATE/COUNTER, but start and stepsize will be different!

**See also:** ROTATE/CLOCK, REBIN/ROTATE

**Examples:** `ROTATE/COUNTER paloma blanca 2`
rotate image paloma.bdf by 180 degrees counter_clockwise and store result into frame blanca.bdf

# RUN

## RUN — *core* — 04-OCT-1991 KB

**Purpose:** Execute a program inside MIDAS and pass over current keywords.

**Syntax:** `RUN progr`

| | |
|---|---|
| `progr` | complete file specification of executable program, if not in current directory; if no file type given, ".exe" is appended; Warning: if no program name is given, MIDAS tries to execute a module with name ?.exe ... |

**See also:** @@, MIDAS Environment document

**Note:** It is assumed, that the program has been written with the FORTRAN or C interfaces from the MIDAS Environment.
With this command you assure correct keyword communication between the program and the monitor.
Beware, in VMS the command $RUN also executes a module:
Using $RUN instead of RUN in MIDAS will execute the module, but the keywords will contain garbage.
For Unix the program name (without path specification!) will be converted to lowercase.

**Examples:** `RUN tecate`
Execute module tecate.exe .

`RUN bohemia.cerv`
Execute module bohemia.cerv .

`RUN ESO$DISK:[ESO]dosequis`
Execute module ESO$DISK:[ESO]dosequis.exe on a VMS machine.

`RUN /esodisk/eso/cartablanca`
Execute module /esodisk/eso/cartablanca.exe on a Unix machine.

## SAVE/REGRESSION — *core* — 12-OCT-1983 JDP

**Purpose:** Save the result of a REGRESSION command in table descriptors. Regression coefficients can be referred to by the user defined name.

**Syntax:** `SAVE/REGRESSION table name`

| | |
|---|---|
| `table` | name of table file |
| `name` | regression name |

**Note:** none

**Examples:** `The following set of commands will compute a new column with`
the results of a polynomial regresion :

`REGRESSION/POLYNOMIAL mytable :Y :X 2`
SAVE/REGRESSION mytable TEST COMPUTE/REGRESSION mytable :YN = TEST

## SCROLL/CHANNEL *core* 30-APR-1990 KB

**Purpose:** Scroll image on given ImageDisplay channel

**Syntax:** `SCROLL/CHANNEL [chanl] [scrolx,scroly]`

`chanl` channel no.; defaulted to currently displayed channel

`scrolx,scroly` fixed scroll values; if omitted, the scroll values are modified interactively

**See also:** CLEAR/SCROLL, DISPLAY/CHANNEL, SHOW/CHANNEL, ZOOM/CHANNEL, LOAD/IMAGE

**Note:** If you work with a DeAnza then:
For interactive scrolling, the cursor box has to be set up with cursor0 on, TRACK off, RATE on. Moving the joystick/trackball will scroll the image. To exit turn cursors off + press ENTER.
If you work with an XWindow system:
For interactive scrolling, use the arrow keys of the keyboard (with the mouse cursor in the display window!). To exit press EXIT button on the mouse.

**Examples:** `SCROLL/CHAN 0 200,200`
Set scroll values of channel 0 to 200,200 .

`SCROLL/CHAN`
Change scroll values of displayed channel interactively.

# SEARCH/FCAT

SEARCH/FCAT *core* 24-JUN-1991 KB

**Purpose:** Search in fit file catalog for frame with matching descriptor IDENT.

**Syntax:** `SEARCH/FCAT [cat_name] search_string [options]`

`cat_name` name of Fit File Catalog;
defaulted to the currently active Fit File Catalog

`search_string` text pattern for which to search

`options` display + searchflag (2 chars.),
1. char. = D or N, for D a message is displayed if a matching frame has been found, else not;
2. char. = B or E, for B search is done from the beginning, else from the end of the catalog.
defaulted to DB

**See also:** SEARCH/TCAT, SEARCH/ICAT
SET/FCAT, CREATE/FCAT, READ/FCAT, ADD/FCAT, SUBTRACT/FCAT

**Note:** The search is done from the beginning or from the end of the catalog and the first frame with a matching IDENT descriptor is returned in keyword OUT_A.
If no match, OUT_A is set to all blanks.
If the search_string contains blanks, it has to be enclosed in double quotes (").

**Examples:** `SEARCH/FCAT lolly FF1024`
Search in Fit File Catalog lolly.cat for a frame with an IDENT descriptor containing the string "FF1024". If found, display relevant message.

`SEARCH/FCAT lolly FF1024 de`
As above but we start searching with the last entry in the catalog and work our way down to the first entry.

`SEARCH/FCAT ? "aa bb" NB`
Search in the currently active Fit File Catalog for a frame with an IDENT descriptor containing the string "aa bb". No message is displayed.

SEARCH/ICAT *core* 24-JUN-1991 KB

**Purpose:** Search in image catalog for frame with matching descriptor IDENT.

**Syntax:** `SEARCH/ICAT [cat_name] search_string [options]`

`cat_name` name of Image Catalog;
defaulted to the currently active Image Catalog

`search_string` text pattern for which to search

`options` display + searchflag (2 chars.),
1. char. = D or N, for D a message is displayed if a matching frame has been found, else not;
2. char. = B or E, for B search is done from the beginning, else from the end of the catalog.
defaulted to DB

**See also:** SEARCH/TCAT, SEARCH/FCAT
SET/ICAT, CREATE/ICAT, READ/ICAT, ADD/ICAT, SUBTRACT/ICAT

**Note:** The search is done from the beginning or from the end of the catalog and the first frame with a matching IDENT descriptor is returned in keyword OUT_A.
If no match, OUT_A is set to all blanks.
If the search_string contains blanks, it has to be enclosed in double quotes (").

**Examples:** `SEARCH/ICAT lolly FF1024`
Search in Image Catalog lolly.cat for a frame with an IDENT descriptor containing the string "FF1024". If found, display relevant message.

`SEARCH/ICAT lolly FF1024 de`
As above but we start searching with the last entry in the catalog and work our way down to the first entry.

`SEARCH/ICAT ? "aa bb" NB`
Search in the currently active Image Catalog for a frame with an IDENT descriptor containing the string "aa bb". No message is displayed.

# SEARCH/LINE

## SEARCH/LINE *core* 22-JUNE-1984 JDP

**Purpose:** Search for emission/absorption lines in spectra.

**Subject:** Spectral analysis, wave calibration.

**Syntax:** `SEARCH/LINE frame w,t[,nscan] [table] [meth] [type]`

| | |
|---|---|
| `frame` | input spectrum |
| `w,t,nscan` | input parameters as:<br>w (window) is the approx. zero intensity line width in pixels.<br>t (threshold) is the absolute value of the background.<br>nscan number of scan lines of the input spectrum to be averaged before searching for the lines. Default is one. |
| `table` | output table. Default name is LINE The output table contains the columns:<br>:X - center of the line<br>:Y - scan number for 2D spectra<br>:PEAK - value at the maximum/minimum |
| `method` | centering method as<br>GRAVITY - center of gravity of the 2 highest pixels with respect to the third one (default for emission lines)<br>MAXIMUM - pixel with the maximum value<br>MINIMUM - pixel with the minimum value (default for absorption lines)<br>GAUSSIAN- center of the gaussian fitted to the line |
| `type` | EMISSION (Default) or ABSORPTION |

**Note:** Lines are detected by means of a local threshold over the background. Lines are centered according to the selected method.

**See also:** CALIBRATE/WAVE, IDENTIFY/LINE, PLOT/IDENTIFY

**Examples:** `SEARCH/LINE SPECTRUM 5,100 LINE GRAVITY EMISSION`
This command will search for emission lines in spectrum having more than five pixels and peak value over 100 in the local background.

SEARCH/TCAT *core* 24-JUN-1991 KB

**Purpose:** Search in table catalog for table with matching descriptor IDENT.

**Syntax:** `SEARCH/TCAT [cat_name] search_string [options]`

`cat_name` name of Table Catalog;
defaulted to the currently active Table Catalog

`search_string` text pattern for which to search

`options` display + searchflag (2 chars.),
1. char. = D or N, for D a message is displayed if a matching table has been found, else not;
2. char. = B or E, for B search is done from the beginning, else from the end of the catalog.
defaulted to DB

**See also:** SEARCH/ICAT, SEARCH/FCAT
SET/TCAT, CREATE/TCAT, READ/TCAT, ADD/TCAT, SUBTRACT/TCAT

**Note:** The search is done from the beginning or from the end of the catalog and the first table frame with a matching IDENT descriptor is returned in keyword OUT_A.
If no match, OUT_A is set to all blanks.
If the search_string contains blanks, it has to be enclosed in double quotes (").

**Examples:** `SEARCH/TCAT lolly FF1024`
Search in Table Catalog lolly.cat for a table with an IDENT descriptor containing the string "FF1024". If found, display relevant message.

`SEARCH/TCAT lolly FF1024 de`
As above but we start searching with the last entry in the catalog and work our way down to the first entry.

`SEARCH/TCAT ? "aa bb" NB`
Search in the currently active Table Catalog for a table with an IDENT descriptor containing the string "aa bb". No message is displayed.

# SELECT/TABLE

## SELECT/TABLE *core* 15-Jul-1992 MP

**Purpose:** Select table entries.

**Syntax:** `SELECT/TABLE table logical-expression`

`table` table name

`logical-expression` FORTRAN-like logical expression allowing the user to define a subtable

**Note:** The subtable will be used by commands that do not modify the table information, namely:
PRINT, PLOT, OVERPLOT, STATISTIC, COPY (input), MERGE (input), PROJECT, READ, REGRESSION
The selection criterion will be reset to the whole table by commands which change the table information:
COMPUTE, SORT, COPY (output), MERGE (output), WRITE
or by the logical expression ALL.
Variables in the expression are columns defined by label or number, the sequence number of the entries in table referred by the name SEQUENCE (abb. SEQ),and the logical variables SELECT and ALL to include the previous select mask or the complete table resp. The reserved name NULL can be used with the relational operators .EQ. and .NE.
Expressions may contain arithmetic, relational and logical operators and functions. The supported arithmetic operators are +, -, *, /, and **. Relational operators are:
.LE. .LT. .GE. .GT. .EQ. .NE.
Logical operators are: .AND. .OR. .NOT.
The supported mathematical functions are:
SQRT(:a) LN(:a) LOG10(:a) EXP(:a) SIN(:a) COS(:a) TAN(:a) ASIN(:a) ACOS(:a) ATAN(:a) SINH(:a) COSH(:a) TANH(:a) ABS(:a) INT(:a) MIN(:a,:b) MAX(:a,:b) MOD(:a,:b)
Angular units of trigonometric functions are degrees.
In selections involving columns with floating numbers one should avoid expressions like :COLUMN.EQ.value; in this case the expression ABS(:COLUMN-value).LT.epsilon is recommended.
One can select entries of a column containing character strings which match a pattern . The following special characters may be used to build these patterns:
* Wildcard character, replaces 0 to N characters
? Replaces 1 character
[] matches n alternative characters
Ignores case
The number of selected rows in the table is stored in the keyword OUTPUTI(1).
The selection flag of a given row of a table may be read and copied into a keyword using the syntax:
KEYWORD = 'table,sel,row'

**See also:** COMPUTE/TABLE

**Examples:** `SELECT/TABLE mytable :MAGNITUDE.GT.7.0`
Select all entries of the table 'mytable.tbl' for which the value of the column MAGNITUDE is greater than 7.

`SELECT/TABLE mytable .NOT.SELECT.AND.:Y.GT.0`
Select all entries of the table 'mytable.tbl' which were not selected in the last SELECT command and for which the value of the column Y is greater than 0.

`SELECT/TABLE mytable ALL`
Select the whole table.

`SELECT/TABLE mytable SEQ.eq.3`
Select 3rd row of table 'mytable.tbl'.

`SELECT/TABLE mytable :STRING.EQ."S*7[ab]"`
Select entries of the table 'mytable.tbl' for which the contents of the column STRING match the following pattern: the char. string should start with S and end with 7a or 7b.

---

SET/ACAT *core* 03-JUN-1991 KB

**Purpose:** Make given catalog the "active" ASCII file catalog.

**Syntax:** `SET/ACAT [cat_name]`

| | |
|---|---|
| `cat_name` | name of catalog which should be enabled; defaulted to 'acatalog.cat' |

**See also:** CLEAR/ACAT, CREATE/ACAT

**Note:** After having enabled an ASCII file catalog, files may either be accessed by their name or by their sequence number in the enabled (= currently active) catalog.
Contrary to image, tabler and fit file catalogs, ASCII files are not automatically added to an active ASCII file catalog.
Main usage of this command is for OUTTAPE/FITS which will write to tape the files in the currently active ASCII file catalog.

**Examples:** `SET/ACAT`
Make the ASCII file catalog 'acatalog.cat' the currently active ASCII file catalog in MIDAS.

`SET/ACAT lola`
Make the ASCII file catalog 'lola.cat' the currently active one.

# SET/BACKGROUND

## SET/BACKGROUND *core* 04-AUG-1994 KB

**Purpose:** Put Midas session into "background" mode

**Syntax:** `SET/BACKGROUND [method] [echo] [sleep_time]`

| | |
|---|---|
| `method` | SOCKETS or FILES, if we want to communicate via Unix sockets or via ASCII files.<br>For the SOCKETS method the communicating Midas sessions may run on the same host or different hosts in the network.<br>If the Midas session from which we will receive commands is not running on the same host , the "SOCKETS,remote" has to be used.<br>For the FILES method the communicating Midas sessions have to run on the same host. |
| `echo` | ON or OFF, to echo the no. of bytes received and the unit of the sending Midas or not;<br>defaulted to ON |
| `sleep_time` | no. of seconds to wait before looking for input;<br>only applicable for 'method'=FILES, defaulted to 1 |

**See also:** CLEAR/BACKGROUND, CONNECT/BACK_MIDAS

**Note:** This command currently works only in Unix, not in VMS!
In background mode Midas does not accept input typed at the keyboard. Instead Midas waits for commands sent by other Midas sessions
If communicating Midas sessions run on the same host, they must be started in parallel mode.
If the communication is based on files, the background Midas receives a signal from the sending Midas when a command has been sent.
The par. 'sleep_time' is only needed for those cases where the sender is not another Midas session but a stand-alone program on a different host executing in a different environment, e.g. OS/9 or "Rocky Mountain Basic" from HP.
If the communication is based on sockets, i.e. the Unix OSX interfaces are used, the SET/BACKGROUND command has to be given before any other Midas connects to this session via the CONNECT/BACK_MIDAS command

**Examples:** `SET/BACKGR`
Put current Midas into background mode, use ASCII files for communication and echo the sender Midas unit for incoming commands.

`SET/BACKGR sockets off`
Put current Midas into background mode, use the osx interfaces communication and do not echo the sender Midas unit for incoming commands.
Only input from Midas sessions running on the same host is received.

`SET/BACKGR sockets,remote off`
As above, but input from Midas sessions running on other hosts in the network is received.

## SET/BUFFER *core* 24-JUN-1991 KB

**Purpose:** Set up (new) size of command buffer for MIDAS.

**Syntax:** `SET/BUFFER [no_lines]`

`no_lines` no. of command lines in buffer; defaulted to 15

**See also:** CLEAR/BUFFER, WRITE/COMMANDS, READ/COMMANDS

**Note:** The no. of lines is currently limited to 40.

**Examples:** `SET/BUFF 22`
Set size of command buffer to 22 lines.

## SET/CONTEXT *core* 27-APR-1987 KB

**Purpose:** Enable new context and increment context level.

**Syntax:** `SET/CONTEXT cntxt`

`cntxt` name of context (max. 8 characters!);
for each context "cntxt" there must exist a file "cntxt.ctx" either in MID_CONTEXT: or in the current directory or in MID_WORK: (that's also the search order)
the context is directly translated into a file name and since by default all filenames are lowercase in MIDAS the context name is always converted to lowercase!

**See also:** CLEAR/CONTEXT, SHOW/CONTEXT, HELP [Context], SHOW/COMMAND
chapter 3, MIDAS Users Guide, volume A

**Note:** This command is used to enable application packages which are not part of the core MIDAS. The context file is a normal MIDAS procedure used to add new commands, to set specific defaults, to copy standard files and so on.
HELP [Context] displays general info about contexts and a list of all contexts currently provided by MIDAS (i.e. context files in MID_CONTEXT).

**Examples:** `SET/CONTEXT echelle`
Enable all the commands used for Echelle type data reduction.

`SET/CONTEXT tijuana`
If you have a procedure tijuana.ctx in the current directory or in MID_WORK:, this procedure will be executed and the context level incremented.

# SET/CURSOR

## SET/CURSOR *core* 24-SEP-1993 KB

**Purpose:** Set cursor form and position.

**Syntax:** `SET/CURSOR [curs_no] [curs_form] [curs_coords] [flag]`

| | |
|---|---|
| `curs_no` | 0, 1 or 2 for cursor 1, 2 or both cursors;<br>defaulted to 0 |
| `curs_form` | defines the cursor(s) form as<br>(a) PROGRAM to use the programmable cursor(s)<br>(b) C_HAIR for large cross hair<br>(c) CROSS,WHITE or CROSS,BLACK for white or black cursor cross<br>(d) OPEN_CROSS,WHITE or OPEN_CROSS,BLACK for white or black open cursor cross<br>(e) ARROW for a white arrow<br>(f) RECTANGLE to use both cursors for a white rectangle<br>(g) CIRCLE to use both cursors for up to 3 circles<br>(h) 0,...,15 to use one of the 16 fixed cursor shapes;<br>only valid for DeAnza;<br>defaulted to PROGRAM |
| `curs_coords` | optional string of cursor coordinates;<br>x,y or x1,y1,x2,y2 depending upon, if one or both cursors are used.<br>For the option (g) above: 'curs_coords' = xcen,ycen,rin,rmid,rout indicating the center x,y coords and radius of the inner, middle, and outer circle. The radii are defined in screen pixels.<br>If curs_coords are omitted, cursors remain where they are; this is the default |
| `flag` | F(RAME), if the cursor coords. in parameter 'curs_coords' have to be interpreted as frame pixels (of the currently displayed image) according to the MIDAS rules,cf. READ/IMA<br>= S(CREEN), the coords in curs_coords are interpreted as screen pixels;<br>defaulted to S |

**See also:** CLEAR/CURSOR, SET/GCURSOR, GET/CURSOR

**Note:** For XWindow systems only cursor form (a), (b), (f) and (g) are implemented. Form (a) yields an open, small cross. All cursor shapes are connected to the mouse.
Form (f) yields a rectangle. The center of the rectangle is moved via the mouse and its size is changed via the 'arrow' keys on the keyboard.
Form (g) yields one, two or three circles depending upon which radius is nonzero (the inner radius must be $> 0$). The center of the circles is moved via the mouse. The size of the circles is changed via the 'arrow' keys on the keyboard. Via the function keys F1, F2, F3 and F4 of the keyboard you specify which circle is affected the next time you hit an 'arrow' key:
F4 enables the updating of the inner circle and the middle and outer circle (if they exist) are changed accordingly. This is the default.
F1 enables the updating of the inner circle only. The radius of the inner circle is forced to at least 1 pixel and cannot be larger than the middle or outer circle.
F2 enables the updating of the middle circle only. The radius of the middle circle is forced to remain larger than the radius of the inner circle and less than the one of the outer circle (if existing).
F3 enables the updating of the outer circle only. The radius of the outer circle is forced to remain larger than the radius of the middle circle (if existing).
The keys 0, 1, ..., 9 control the increments of the changing of size.
Also, the colour of the cursor is not strictly white but depends on the underlying pixel colour...
For a demonstration of the different available cursor forms on the DeAnza Image display use the MIDAS command TUTORIAL/CURSOR.
Form (g) is not supported for DeAnzas.
The effect of setting the cursor to a specified value will be seen only in a subsequent command which uses the cursor.

**Examples:** `SET/CURSOR ? RECTANGLE`
Use white cursor rectangle, specify no position for its corners.

`SET/CURSOR 0 ?  @100,@212 F`
Use currently loaded cursor shape with cursor 0, specify its position as the one where frame pixels 100,212 of the displayed frame are located
If not a DeAnza display, only the next command which uses a cursor will show the cursor in the corresponding position.

`SET/CURSOR ? circle 200,200,10,20,30`
Define cursor circles. Initial position will be centered at screen pixels (200,200) and the radii of the inner, middle and outer circle will be 10, 20 and 30 pixels,respectively.

# SET/DISPLAY

## SET/DISPLAY *core* 17-OCT-1983 KB

**Purpose:** Set up Image Display for RGB or pseudo colours.

**Syntax:** `SET/DISPLAY [colour_mode]`

`colour_mode` RGB, to set up Image Display in RGB mode -
then the channels 0,1 and 2 hold the red, green and blue component of an image and are displayed together for true colour
or PSEUDO, for pseudo colours;
defaulted to PSEUDO

**See also:** CLEAR/DISPLAY
chapter 6, MIDAS Users Guide, volume A

**Note:** This command is only implemented for DeAnza image displays.

**Examples:** `SET/DISPLAY PSEUDO`
set DeAnza image display to pseudo colour mode for each channel

## SET/FCAT *core* 16-JAN-1990 KB

**Purpose:** Make given catalog the "active" fit file catalog.

**Syntax:** `SET/FCAT [cat_name]`

`cat_name` name of catalog which should be enabled, defaulted to 'fcatalog.cat'

**See also:** CLEAR/FCAT, CREATE/FCAT

**Note:** After having enabled a fit file catalog, fit files may either be accessed by their name or by their sequence number in the enabled ( = currently active) catalog.
Also for each newly created fit file an entry is added to the active fit file catalog.

**Examples:** `SET/FCAT`
Make the MIDAS fit file catalog 'fcatalog.cat' the currently active fit file catalog.

`SET/FCAT lola`
Make the MIDAS fit file catalog 'lola.cat' the currently active fit file catalog.

SET/FORMAT *core* 15-SEP-1995 KB

**Purpose:** Set formats for replacement of Midas data, i.e. keywords, table elements, image pixels or descriptors.

**Syntax:** `SET/FORMAT [format_specs]`

`format_specs` formats for integer or real,double data and may be of the form:
Ix for integer values (x in [1,9]);
Fxx.yy or Exx.yy (Gxx.yy) for real, double values (x,y in [1,9]);
formats for real and double data are specified in the same format string (parameter), separated by a comma:
use one format without a comma to specify a format for real data only and use ',format' to specify a format for double data only

**See also:** Chapter 3 of MIDAS Users Guide, Volume A

**Note:** Except the I-format these formats are processed like the corresponding FORTRAN Format statements;
the I format prints also leading zeroes, and if the number does not fit into the format it is printed anyway (so no '***' output as in Fortran will occur).
One or two formats can be specified in one command line. If only one format is specified the other one is left unchanged. Thus,
SET/FORMAT I6
only changes the format for integer data and leaves the formats for real and double data as they are.
If no format specification is given at all, the default formats I4 E15.5,E15.5 are applied, i.e. entering SET/FORMAT is the same as entering SET/FORMAT I4 E15.5,E15.5 .
The formats are preset to I4, E15.5 and E15.5 when starting Midas. For backward compatibility the double format is set equal to the real format, but, e.g. format E22.15 would be more appropriate for double precision data.
The format specifications are NOT global but linked to the procedure level. Thus, if you specify a format in procedure abc.prg and call another procedure xyz.prg from procedure abc.prg, the formats used in xyz.prg will be the default formats - not the ones specified in abc.prg at the higher level.
Please, note, that these format specifications are not only used for displaying data (WRITE/OUT command) but also whenever a data element is enclosed in curly brackets in a Midas procedure!

**Examples:** `SET/FORMAT I5 E12.6`
Will yield: -01234 -1.234560E+02 -1.23457E+05
when using the Midas command:
write/out "inputi(1) inputr(1) inputd(1)"
assuming that INPUTI(1) = -1234, INPUTR(1) = -123.456 and INPUTD(1) = -123456.98765432

`SET/FORMAT ,f30.10`
Will yield: -1234 -1.23456E+02 -123456.98765432
only the double format has been modified.

`SET/FORMAT f12.7,f30.10 i1`
Will yield: -1234 -123.4560013 -123456.98765432
Note, that the real format is "too large", so random digits are added to the output of INPUTR.
Also, the full integer value is displayed, so 'I1' is a safe format for all integers (see the Note above).

`SET/FORMAT`
Will yield: -1234 -1.23456E+02 -1.234569876543200E+0
the default formats.

# SET/GCURSOR

```
COMPUTE/IMAGE guerrero = chiapas*(inputr(1))
```
Will result in the execution of the command:
COMPUTE/IMAGE guerrero = chiapas*(-1.23456E+02)

---

SET/GCURSOR *core* 07-JAN-1991 KB

**Purpose:** Set cursor form in graphics window.

**Syntax:** `SET/GCURSOR [curs_no] [curs_form]`

| | |
|---|---|
| `curs_no` | currently not supported, always enter '?' |
| `curs_form` | defines the cursor(s) form as<br>(a) CROSS for cursor cross, the default<br>(b) C_HAIR for large cross hair<br>(c) OPEN_CROSS for cursor cross with open center |

**See also:** GET/GCURSOR

**Note:** none

**Examples:** `SET/GCURSOR`
Use small cursor cross.

`SET/GCURSOR ? C_HAIR`
Use large crosshair cursor.

SET/GRAPHICS *core* 15-JUL-1988 RHW

**Purpose:** Set plot characteristics.

**Subject:** Graphics.

**Syntax:** `SET/GRAPHICS option1[=value1] option2[=value2] ...`

| | |
|---|---|
| `option` | plot characteristics to be set (value). The following options with their values are possible:<br>-1: data only; 0: data and a coordinate box; 1: data, a ccordinate box and minimal legend; 2: data and full legend. |
| `DEFAULT` | no value; sets plot package in default mode (below) |
| `XAXIS` | AUTO or xstart,xend,xbig,xsmall in world coordinates; define the x-axis labelled from xstart to xend. If xbig > 0 use that for the spacing of the large ticks and try to use xsmall for the small ticks. If xsmall < 0 make a logarithmic axis; if xsmall is not given xsmall = xbig; for xsmall = xbig =0 the default values for the ticks are used; default AUTO |
| `YAXIS` | AUTO or ystart,yend,ybig,ysmall in world coordinates; for explanation see above (XAXIS) |
| `ZAXIS` | AUTO or zstart,zend,zbig,zsmall in world coordinates; for explanation see above (XAXIS) |
| `FRAME` | REctangular or SQuare to set the coordinate frame (box) in rectangular (default) of square mode. If this option is used previous setting(s) of the XSCALE and/or YSCALE value(s) will be overwritten |
| `XSCALE` | AUTO or xscale: scale or size of the plot in the x direction. A positive number is interpreted as scale parameter (world units/mm); a negative number as size parameter (axis will be made that among of mm. long). The AUTO setting will make a plot that fills the available space in the x direction. If this option is used the previous setting of the FRAME option will be overwritten (set to "RECT") |
| `YSCALE` | AUTO or yscale. For explanation see above (XSCALE) |
| `XOFFSET` | NONE, or xoffset: the distance in mm of the left y axis of the plot to the left boundary of the graphic window/device. The default value NONE will place the plot at the left of the graphics device |
| `YOFFSET` | NONE, or yoffset: the distance in mm of the lower x axis of the plot to the lower boundary of the graphic window/device. The default value NONE will place the plot at the top of the graphics device |

# SET/GRAPHICS

**XFORMAT** NONE, AUTO, or format. Format specification to draw the tickmark labels. The string can contain some of the standard C or FORTRAN format specification as well as a set of special format specifications listed below.
For the standard notation w defines the minimum width of the field, p the number of decimal positions. Format strings between square brackets are defaulted.
Standard formats are:
[w.p]d (C): decimal integer (comp. F77 Iw.d);
[w.p]f (C): floating point (comp. F77 Fw.d);
[w.p]e (C): exp. format (comp. F77 Ew.d);
I[w.p] (F): decimal integer (eg I6.3 = 6.3d);
F[w.p] (F): floating point (eg F8.5 = 8.5f);
E[w.p] (F): exp. format (eg E15.3 = 15.3e);
Special formats are:
g (S): as f format, but strips all
zeroes and decimal point
p (S): write only the power of 10;
x (S): write only the powers of e;
[a/h][u][m][s][s][s]...: astronomical notation,
where:
[a/h] : degrees (a) or hours (h);
[u] : units;
[m] : minites;
[s] : seconds;
[s] : first digit of fraction;
[s] : second digit of fraction;
[s] : etc...

With AUTO you will obtain (hopefully) an axis with sensible tickmarks. In case the format is set to NONE no tickmark label will be written. Default is AUTO.

**YFORMAT** NONE, AUTO or format description; see XFORMAT

**ZFORMAT** NONE, AUTO or format description; see XFORMAT

**PMODE** Plot mode in which the plot package will run. Depending on the PMODE the graphics package writes additional information on the graphics output; -1 – plot without frame and legend; 0 – plot with data and axes only; 1 – plot with data, frame and some information; 2 – plot with frame and full legend; default is 2

**FONT** Font to be used to write text in the graphic output: 0 – Standard simple default font; 1 – Roman font; 2 – Greek font; 3 – Script font; 4 – Old English font; 5 – Tiny roman font; default is 0.

**LTYPE** number: 0 – no line at all ; 1 – solid; 2 – dotted; 3 – short dash; 4 – dash - dot; 5 – long dash; 6 – dash - dot - dot; default is 1

**LWIDTH** n ranging from 0 to 4; 0 or 1 corresponds with single width, 2 with double width, etc. Default is 0

**STYPE** number: 0 – no symbol; 1 – dot; 2 – hexagon; 3 – square; 4 – triangle; 5 – cross (+); 6 – cross (x); 7 – asterisk; 8 – star; 9 – crossed square (+); 10 – crossed square (x); 11 – lozenge; 12 – hor. bar; 13 – vert. bar; 14 – right arrow; 15 – arrow up; 16 – left arrow; 17 – arrow down; 18 – filled hexagon; 19 – filled square; 20 – filled triangle; 21 – filled lozenge; A symbol can also be a character as defined in the decimal ASCII character set: numbers between 33 - 126 48 – 0; 65 – A; 97 – a; Default is 5 (cross: +)

| | |
|---|---|
| SSIZE | number: symbol size multiplication factor; default is 1 |
| TSIZE | number: character size multiplication factor; default is 1 |
| TWIDTH | integer number: set line width for text strings. This feature is not supported on all graphics devices. Higher values will increase the thickness if the characters. This option is supported for a limited number of devices (e.g. postscript printer, NOT for graphics windows). Default is 1 |
| BINMODE | OFF or ON: mode in which lines will be drawn between the data points. OFF will connect the points with a simple straight line; ON will connect the points in a histogram-like manner. This mode will ONLY work for equally spaced data. Hence, table data are excluded and will always be drawn with BINMODE=OFF; default is OFF |
| COLOUR | integer number: ranging from 0 to 7; the colour black is the default (1). This setting is only effective on image display devices (like workstations). The available colours are: default background (0) black (1), red (2), green (3), blue (4), yellow (5), magenta (6) cyan (7), white (8). Default colour is 0, the default background collour, currently white |
| COLMODE | SUB or XOR: set the graphic write mode. In case of SUB the colour written on the graphics device will simply substitute the background colour (overwrite). In case of XOR the system will adjust the colour depending on the background. Default is SUB |
| BCOLOUR | integer number: ranging from 0 to 8. Sets background as specified colour. The default background is white: 0. This option is hardware dependent and is effective only on image display devices (like the graphics window on workstations, and colour laser printers. The new background colour will only be activated after a PLOT command in which the display is cleared first. For the colour available, see above (COLOUR) |
| CLEARGRA | OFF or ON: clear the graphics device before a new plot is produced. This option will only work for real PLOT commands; OVERPLOT commands are NOT affected. Using this option in combination with scale and offset settings, one can plot a number of graphs on the graphics device with a relatively small number of commands. Default is ON: graphics device will be cleared at every PLOT command |

# SET/GRAPHICS

**Note:** The format for astronomical coordinates notations does not work properly. The bug will be fixed. Also, the automatic format adjustment is not working.

When in mode BINMODE OFF the user wants to plot a line of an image or a descriptor, the plot package first looks for the line type. If the line type is set to 0 (LTYPE=0) it looks for a symbol (STYPE). If both LTYPE and STYPE are set to 0 a warning is issued. In cases of histogram plotting or when the bin mode is on (BINMODE=ON) the package assumes a line type greater than zero; an error occurs when LTYPE=0.

In the case of table plotting the package first looks for the symbol type. If STYPE is 0 (STYPE=0) a line will be drawn corresponding with LTYPE. A warning message is given when both LTYPE and STYPE are set to 0.

In case one plots data using a line type not equal 1, the (dotted or dashed) line may look irregular. This can be due to rounding errors or crowding of the (not equally spaced) data points.

The option COLMODE=XOR does not always work really well when COLOUR and BCOLOUR both have specified the same COLOUR.

The scales of a plot may change if a plot is sent to a device other than the original one (pre-specified by ASSIGN/GRAP). In general the axis ratio of the frame will have changed, and hence a square frame WILL NOT BE A SQUARE FRAME ANYMORE if you use COPY/GRAPHICS to dump your plot on another device.

Special character like "+", "-","(" cannot be printed using the fonts 2 and 4 (Greek and Old English). In additional to the various fonts, text can be drawn with a number of options. For example, to write formulae in the graphics output, to use mathematical and astronomical symbols, etc. Also, "LaTeX/TeX like" symbols can be used. For reference have a look at The MIDAS User's Guide, Volume A, Chapter 6.

The graphics setup is stored in a number of special keywords: PLISTAT, PLRSTAT, PLCSTAT, PLCMETA, PLCDATA, and PLRGRAP. For more information about the graphics and these keywords read the MIDAS Environment document or use the command HELP/KEYWORD.

**See also:** SHOW/GRAPHICS, LABEL/GRAPHICS, Chapter 6 of the the MIDAS User's Guide, Vol. A.

**Examples:** `SET/GRAPHICS XAXIS=4000,1000,500,100`
set frame of x-axis manually from 4000 to 1000 with negative increment of 500

`SET/GRAPHICS xax=0,24,6,3 xfo=hu yax=-10,-8,.3,.1 yfo=aums`
set frame of x-axis manually from 0 to 24 hours with a ticks mark distance of 6 hours and small ticks every 3 hours. In the y direction the graph will run from -10 degrees to -8 degrees with ticks every 18 arcmin and small ticks every 6 arcsec.

`SET/GRAPHICS XAXIS YAXIS=-2,2,1,-1 YFORM=p`
set frame of y-axis manually in logarithmic mode from 0.01 to 100 with increment of one order of magnitude. Use the power ten notation for the tick marks labels. Set the default range and steps for the x-axis; for the x-axis the graphic package will find sensible values (I hope).

`SET/GRAPHICS LTYPE=1 BINMODE=ON`
set the line type to 1 (solid line) and plot in bin mode

## SET/ICAT *core* 16-JAN-1990 KB

**Purpose:** Make given catalog the "active" image frame catalog.

**Syntax:** `SET/ICAT [cat_name]`

| | |
|---|---|
| `cat_name` | name of catalog which should be enabled, defaulted to 'icatalog.cat' |

**See also:** CLEAR/ICAT, CREATE/ICAT

**Note:** After having enabled an image catalog, image frames may either be accessed by their name or by their sequence number in the enabled (= currently active) catalog.
Also for each newly created image frame an entry is added to the active image catalog.

**Examples:** `SET/ICAT`
Make the MIDAS image catalog 'icatalog.cat' the currently active image catalog.

`SET/ICAT lola`
Make the MIDAS image catalog 'lola.cat' the currently active image catalog.

## SET/ITT *core* 17-OCT-1983 KB

**Purpose:** Enable ITT for memory channel

**Syntax:** `SET/ITT [chanl] [sect]`

| | |
|---|---|
| `chanl` | Image display memory (or channel), defaulted to currently displayed channel |
| `sect` | ITT section no. (0 - 3) - only possible for DeAnza image displays, for all other devices this parameter is always set to 0, which is also the default |

**See also:** CLEAR/ITT, LOAD/ITT, MODIFY/ITT, EQUALIZE/HISTOGRAM

**Note:** If an ITT (Intesity Transfer Table) is loaded (and enabled) to an image channel, all pixel data in that channel serve as indices into that table and the value at that ITT element is displayed and passed on to the LUT. For more details see Chapter 6 of the MIDAS Users Guide.

**Examples:** `SET/ITT 0`
Enable ITT for channel 0.

# SET/LUT

| SET/LUT | *core* | 14-APR-1986 KB |
|---|---|---|

**Purpose:** Set section of colour lookup table (LUT).

**Syntax:** `SET/LUT [sect]`

| | |
|---|---|
| `sect` | section of LUT - only possible for DeAnza image displays, for all other devices this parameter is always set to 0, which is also the default |

**See also:** CLEAR/LUT, LOAD/LUT, DISPLAY/LUT, MODIFY/LUT

**Note:** For DeAnza image devices 4 different LUTs may be loaded at the same time into the device, where they are called LUT sections, with section = 0,1,2 or 3, after that you can switch rapidly among these LUTs via SET/LUT.
XWindow systems usually have only 1 LUT (exception is the Stellar Machine where we also implemented 4 LUTs!)
For more details about LUTs see Chapter 6 of the MIDAS Users Guide.

**Examples:** `SET/LUT`
Enable use of LUT which is loaded into LUT-section 0

`SET/LUT 2`
use LUT loaded into LUT-section 2

SET/MIDAS_SYSTEM *core* 19-SEP-1995 KB

**Purpose:** Set different modes and options (max. 8 per command) for Midas.

**Syntax:** `SET/MIDAS_SYSTEM option=value`

# SET/MIDAS_SYSTEM

`option` ENVIRONMENT, PROMPT, USER_LEVEL, KEYWORDS,
COMMANDS, EDITOR, OUTPUT, DSCCOPY, CLOVERLAY,
DQ, INSERT_MODE,
PATH, DPATH, EPATH
for ENVIRONMENT (a), value = env_string;
with env_string = Host, Midas, MidHost
to move to the host system, Midas or a combination of both;
host system is VAX/VMS DCL environment or the UNIX shell
for PROMPT (b), value = prompt_string;
with prompt_string = a string of max. 16 characters
for USER_LEVEL (c), value = level[,mode];
with level = NOVICE, for first-time users of MIDAS,
= USER, for "normal" users,
= EXPERT, for experts of MIDAS,
mode = PROMPT, for prompting for all parameters and
showing their defaults
= NOPROMPT, Midas only prompts for non-defaulted parameters, the default.
for KEYWORDS (d), value = glbno,locno;
with glbno = no. of global keywords,
locno = no. of local keywords;
for COMMANDS (e), value = comno,qualno;
with comno = no. of commands,
qualno = no. of qualifiers;
for EDITOR (f), value = name of the preferred editor for commands like e.g. REPORT/PROBLEM;
defaulted to 'vi' for Unix and 'EDT' for VMS
for OUTPUT (g), value = YES, NO or LOGONLY,
to indicate if the output from MIDAS commands should be displayed or not (or just logged);
defaulted to YES
for DSCCOPY (h), value = YES or NO to indicate if all descriptors of an input frame should be copied to the result frame in MIDAS commands;
defaulted to YES
for CLOVERLAY (i), value = YES or NO to indicate if the overlay channel should be cleared or not at the start of commands like EXTRACT/TRACE, CENTER/GAUSS, ...
defaulted to YES
for DQ (j), value = YES or NO to indicate if double quotes (") enclosing a string are included or stripped off in commands WRITE/KEY, WRITE/DESCR and WRITE/FILE commands;
defaulted to NO
for INSERT_MODE (k), value = YES or NO to indicate if we use InsertMode or not when editing Midas commands. That's like hitting Â (CntrlA) while editing a command;
defaulted to NO;
this option only applicable for VMS systems
for PATH (l), value = string of directory(ies) used in locating the Midas procedures for the @@ command (if not in the current directory) - similar to the PATH variable in Unix;
defaulted to MID_WORK
for DPATH (m), value = string of directory(ies) used in locating the Midas data files (if not in the current directory)
DPATH=?? only updates the data path of the IDI server (needed for LOAD/IMAGE)

for EPATH (n), value = string of directory(ies) used in locating the Midas executables (program modules) used with a Midas RUN command (if not in the current directory);
defaulted to MID_WORK

**See also:** @a showmidas (HELP/APPLIC)

**Note:** (a) In the Host environment, all input is passed directly to the host operating system. Except the command SET/MIDAS, which will bring you back into Midas. The Midas prompt is terminated by '$'. All Midas command editing facilities are still available.
In the MidHost environment all commands which are not recognized by Midas are passed on to the host system. Thus you can avoid the '$' in front of many Unix commands like "more", "ls". It also works for VMS (with commands like e.g. "DIRECTORY"). The Midas prompt is terminateby d ':'.
Note, that this environment setting is only applicable to the interactive input, i.e. in procedures you must enter the '$' for a host command.
(c) the user level affects the the following areas of the MIDAS system:
1) length of error messages - NOVICE and USER will yield exhaustive error messages; EXPERT will yield just a one-line message.
2) If working with a DeAnza image display, the system will be completely initialized (with corresponding waiting time) for NOVICE and USER level, not for EXPERT level. So if you put into your Midas login procedure login.prg the command SET/MIDAS USER=EXPERT
you will not experience the image display initialization when starting with Midas.
3) In the prompt mode you will be prompted for each parameter and the current default is displayed. If default is '?' you must enter a value, otherwise just hit RETURN if you want to use the defaults.
(d) Keyword data base will only be expanded if new values exceed current ones.
(e) Command data base will only be expanded if new values exceed the current ones. Use the command "SHOW/COM -X" to get the current values, there CMAX and QMAX show the max. no. of commands and qualifiers.
(h,i,j,k) anything else but YES is interpreted as NO;
(l,m,n) the string may be either: 'dir_path—dir_path—...—dir_path' to set the different dir_specs or '+dir_path—dir_path—...—dir_path' to add more dir_specs to the currently set paths
max. 4 directories may be specified
if 'PATH=' is entered the default path ($MID_WORK) is taken
if 'DPATH=' is entered all data path names are cleared
if 'EPATH=' is entered the default path ($MID_WORK) is taken
If you enter option=? the current option/status is displayed; (if applicable that value is also stored in keyword OUTPUTI or OUTPUTC)
Your private login.prg file in $MID_WORK is the right place for storing all those SET/MIDAS_SYSTEM commands...

**Examples:** `SET/MIDAS USER=EXPERT`
Declare yourself as a MIDAS expert.

`Midas 045> SET/MIDAS PROMPT=Astro`
Then the next command line will prompt: Astro 046>

`SET/MIDAS us=exp pro=Astro`
Same as the two examples above.

`SET/MIDAS key=?`
Display the current max. no. of global and local keywords.

`SET/MIDAS output=no`
Suppress all output from MIDAS commands.

`SET/MIDAS COMMANDS=300,500`
Expand command data base to hold 300 commands + 500 qualifiers, assuming these values are greater than the current ones.

# SET/OVERLAY

`SET/MIDAS PATH=+$HOME/test`
Add '$HOME/test' as a directory to search for procedures when executing the @@ command.

`SET/MIDAS DPATH=$HOME/data/echelle`
Specify '$HOME/data/echelle' as directory to search for Midas data files if they are not found in the current working directory.

---

## SET/OVERLAY *core* 17-OCT-1983 KB

**Purpose:** Enable graphics overlay plane in image display.

**Syntax:** `SET/OVERLAY`

**See also:** CLEAR/OVERLAY, CLEAR/CHANNEL OVERLAY
Chapter 6, MIDAS Users Guide, Volume A

**Note:** Plots in the image display may be drawn in the graphics/overlay plane of an image display instead of the image memory itself.
Enabling the overlay will show the plot on top of the currently displayed image

**Examples:** `SET/OVERLAY`
Display any graph stored in the overlay plane on top of image.

---

## SET/REFCOLUMN *core* 17-OCT-1983 JDP

**Purpose:** Defines a column as reference.
The values in the reference column can be used to access the table.

**Syntax:** `SET/REF table column`

| | |
|---|---|
| `table` | table name |
| `column` | column to be selected as reference. To reset the reference to the sequence number, use SEQUENCE as parameter |

**See also:** COPY/TT

**Note:** none

**Examples:** `SET/REF mytable :DHNUMBER`
Use column labelled :DHNUMBER as reference in table mytable.tbl.

## SET/SPLIT *core* 17-OCT-1983 KB

**Purpose:** Enable split screen

**Syntax:** `SET/SPLIT [chanls]`

`chanls` optional channel no.s, if only 2 channels are to be displayed on the screen:
1,0 for 1 ! 0 ; 2,3 for 2 ! 3
1,2 for 1/3 ; 0,3 for 0/3
defaulted to complete split screen, i.e. 4 channels

**See also:** CLEAR/SPLIT, TUTORIAL/SPLIT
Chapter 6, MIDAS Users Guide, volume A

**Note:** This command is only implemented for DeAnza display systems!
The screen of the Deanza monitor may be divided into four segments with channel 0 in right upper corner, channel 1 in left upper corner, channel 2 in left lower corner and channel 3 in right lower corner, i.e.
1 ! 0
−+−
2 ! 3 .
Each channel may be displayed with a different lookup table.

**Examples:** `SET/SPLIT 1,0`
Display channel 1 on the left side of the display device and channel 0 on the right side.

## SET/TCAT *core* 16-JAN-1990 KB

**Purpose:** Make given catalog the "active" table file catalog.

**Syntax:** `SET/TCAT [cat_name]`

`cat_name` name of catalog which should be enabled, defaulted to 'tcatalog.cat'

**See also:** CLEAR/TCAT, CREATE/TCAT

**Note:** After having enabled a table catalog, table files may either be accessed by their name or by their sequence number in the enabled (= currently active) catalog.
Also for each newly created table file an entry is added to the active table catalog.

**Examples:** `SET/TCAT`
Make the MIDAS table catalog 'tcatalog.cat' the currently active table catalog.

`SET/TCAT lola`
Make the MIDAS table catalog 'lola.cat' the currently active table catalog.

# SHIFT/IMAGE

## SHIFT/IMAGE *core* 16-JAN-1991 KB

**Purpose:** Shift the pixels in an image in x and y-direction.

**Syntax:** `SHIFT/IMAGE inframe outframe [x,yshift]`

| | |
|---|---|
| `inframe` | name of input image |
| `outframe` | name of output (shifted) image |
| `x,yshift` | no. of pixels to shift in x and in y-direction; defaulted to 1,0 (one pixel in x, no pixel in y) |

**See also:** FLIP/IMAGE

**Note:** Pixels (lines) are shifted right (towards the high end) and wrapped around. To shift left use negative shifts.

**Examples:** `SHIFT/IMAGE mata hari 0,12`
Shift the lines of frame 'mata.bdf' 12 lines up, the 12 top lines will be shifted to the bottom. The result is stored in the frame 'hari.bdf' .

## SHOW/ACAT *core* 03-JUN-1991 KB

**Purpose:** Display the total no. of entries + last entry no. in an ASCII file catalog.

**Syntax:** `SHOW/ACAT [cat_name] [display_flag]`

| | |
|---|---|
| `cat_name` | name of ASCII file catalog; defaulted to currently active ASCII file catalog |
| `display_flag` | D(isplay) or N(odisplay), for displaying the information on the terminal or not; defaulted to D |

**See also:** READ/ACAT, SUBTRACT/ACAT, ADD/ACAT
chapter 3, MIDAS Users Guide, volume A

**Note:** The total no. of entries + last entry no. are also written into integer keyword OUTPUTI(1,2). See the help of the SET/ACAT command to find out how to make a catalog "active"

**Examples:** `SHOW/ACAT vinchuga`
Display the no. of entries + last entry no. in ASCII file catalog vinchuga.cat

## SHOW/BACK_MIDAS *core* 18-JAN-1991 KB

**Purpose:** Show info related to background Midas sessions.

**Syntax:** `SHOW/BACK_MIDAS [option]`

| | |
|---|---|
| `option` | C for showing all info related to previous CONNECT/BACK command;<br>= B for showing all current background Midas sessions;<br>defaulted to B |

**Note:** Currently up to 4 background sessions may be active in parallel.

**See also:** CONNECT/BACK

**Examples:** `SHOW/BACK C`
Show all "background characters" set up with CONNECT/BACK so far.

## SHOW/CHANNEL *core* 17-OCT-1983 KB

**Purpose:** Display all relevant information like number of screen pixels, scroll values, etc. for given channel.

**Syntax:** `SHOW/CHANNEL [chanl]`

| | |
|---|---|
| `chanl` | ImageDisplay channel (image memory plane);<br>defaulted to currently displayed channel |

**See also:** DISPLAY/CHANNEL, SHOW/DISPLAY

**Note:** none

**Examples:** `SHOW/CHAN 1`
Display info about channel 1.

## SHOW/CODE *core* 18-DEC-1990 KB

**Purpose:** Display the procedure which implements the command_string.

**Syntax:** `SHOW/CODE command_string [flag]`

| | |
|---|---|
| `command_string` | idas command/qualifier |
| `flag` | O or T for original or translated code; defaulted to O |

**See also:** SHOW/COMMAND

**Note:** none

**Examples:** `SHOW/CODE load/ima`
Display the Midas procedure which implements the LOAD/IMAGE command.

# SHOW/COMMAND

## SHOW/COMMAND *core* 24-JAN-1992 KB

**Purpose:** Show the actual procedure which implements a MIDAS command.

**Syntax:** `SHOW/COMMAND [comnd/qualif]`

`comnd/qualif` command/qualifier combination of a MIDAS command; if omitted, all user defined commands are displayed

**See also:** SHOW/CODE, HELP command/qualif, HELP/QUALIF

**Note:** Also the default qualifier for the given command is shown as well as the default flag, which is 1 or 0 depending upon if the command CREATE/DEFAULTS had been used for that command to enforce user specific default values.
If instead of a command/qualifier you enter -D (for diagnostics) the internal structures of the command database and their size are displayed.
If instead of a command/qualifier you enter -X only the size of the internal structures of the command database are displayed.

**Examples:** `SHOW/COM LOAD/IMA`
Display relevant info for MIDAS command LOAD/IMAGE.

`SHOW/COM`
Display all user defined commands.

`SHOW/COM -x`
Display size and usage of internal command database.

## SHOW/DEFAULTS *core* 19-FEB-1992 KB

**Purpose:** Display all currently defined special default values and their corresponding commands.

**Syntax:** `SHOW/DEFAULTS`

**See also:** CREATE/DEFAULTS, DELETE/DEFAULTS

**Note:** All defaults are cleared when terminating a MIDAS session via 'bye'. It is, therefore, good practice to put all default setings into the personal login.prg file.

**Examples:** `SHOW/DEF`
Display all currently exisiting special default settings.

## SHOW/DESCR *core* 05-DEC-1991 KB

**Purpose:** Display name, type and size of descriptors of given frame.

**Syntax:** `SHOW/DESCR frame [dsclist] [flag]`

| | |
|---|---|
| `frame` | name of image |
| `dsclist` | list of descr. names separated by comma or wildcard (*);<br>defaulted to '*', i.e. all descriptors of frame |
| `flag` | H(idden) if you only want descr. info in keyword OUTPUTI (see Note) and no display;<br>defaulted to N(ohidden) |

**See also:** READ/DESCR, PRINT/DESCR, WRITE/DESCR, DELETE/DESCR

**Note:** If dsclist is set to a specific descr. name (not the '*'), also the elements 1,2,3,4 of integer keyword OUTPUTI are filled:
OUTPUTI(1) = 1 or 0, if descr. exists or not.
If OUTPUTI(1) is 1, then (2) = descr. type (1,2,3 or 4 for integer, real, character or double descr); (3) and (4) hold no. of elements and bytes per element of descr.
This information can then be used in a procedure.

**Examples:** `SHOW/DESC durazno`
Show all the descriptors of frame 'durazno.bdf'.

`SHOW/DESC pera.tbl`
Show all the descriptors of table file 'pera.tbl'.
Note, that we have to specify the full file name if it is not an image file.

`SHOW/DESC manzana BugsBunny`
Show type and size of descriptor BugsBunny of frame 'manzana.bdf' if it exists. The keyword OUTPUTI will be set accordingly.

## SHOW/DISPLAY *core* 12-SEP-1995 KB

**Purpose:** Show current status of Display and GraphicsWindow.

**Syntax:** `SHOW/DISPLAY`

**See also:** SET/DISPLAY, SHOW/CHANNEL

**Note:** The elements 1, ..., 14 of integer keyword OUTPUTI are filled with the displayed values as follows:
No. of active display, x-dim, y-dim, depth of display, (1 - 4)
No. of image channels, x-dim, y-dim, depth of channels, (5 - 8)
active image channel, No. of LUTs, size of LUT, (9 - 11)
No. of active graphics, x-dim, y-dim of graphics window. (12 - 14)
If no display w. active, OUTPUTI(1) = -1;
if no graphics w. active, OUTPUTI(12) = -1.

**Examples:** `SHOW/DISPLAY`
Display status of ImageDisplay and GraphicsWindow.

# SHOW/FCAT

## SHOW/FCAT *core* 10-OCT-1988 KB

**Purpose:** Display the total no. of entries + last entry no. in a fit file catalog.

**Syntax:** `SHOW/FCAT [cat_name] [display_flag]`

`cat_name` name of fit file catalog;
defaulted to currently active fit file catalog

`display_flag` D(isplay) or N(odisplay), for displaying the information on the terminal or not;
defaulted to D

**See also:** READ/FCAT, SUBTRACT/FCAT, ADD/FCAT
Chapter 3, MIDAS Users Guide, Volume A

**Note:** The total no. of entries + last entry no. are also written into integer keyword OUTPUTI(1,2).
See the help of the SET/FCAT command to find out how to make a catalog "active"

**Examples:** `SHOW/FCAT vinchuga`
Display the no. of entries + last entry no. in fit file catalog vinchuga.cat

## SHOW/GRAPHICS *core* 09-JUN-1987 RHW

**Purpose:** Show the setup parameters for the plotting package.

**Subject:** Graphics

**Syntax:** `SHOW/GRAPHICS`

**Note:** SHOW/GRAPHICS displays the current setting of the graphic package, grouped as follows:
graphics device and plot file;
axes, scales, tickmarks;
plot mode, fonts, symbols, lines, etc ...

In de first group the user can find which graphics device is currently assigned (Device:), which data has plotted last with a PLOT/ command (see below), and which plot file is currently active.

The name of the data file, together with the coordinates specs. (world coordinates, pixel coordinates) of the LAST PLOT command (NOT overplot command) are kept in the system. The GET/GCURSOR command will use this information to retrieve information from graph displayed.

For the meaning of the other parameters consult the documentation of the SET/GRAPHICS command.

The graphics setup is stored in a number of special keywords: PLISTAT, PLRSTAT, PLCSTAT, PLCMETA, PLCDATA, and PLRGRAP. For more information about the graphics and these keywords read the MIDAS Environment document or use the command HELP/KEYWORD.

**See also:** SET/GRAPHICS, GET/GCURSOR, Chapter 6 of the MIDAS User's Guide (Volume A)

**Examples:** `SHOW/GRAPHICS`

## SHOW/ICAT *core* 10-OCT-1988 KB

**Purpose:** Display the total no. of entries + last entry no. in an image catalog.

**Syntax:** `SHOW/ICAT [cat_name] [display_flag]`

| | |
|---|---|
| `cat_name` | name of image catalog;<br>defaulted to currently active image catalog |
| `display_flag` | D(isplay) or N(odisplay), for displaying the information on the terminal or not;<br>defaulted to D |

**See also:** READ/ICAT, SUBTRACT/ICAT, ADD/ICAT
chapter 3, MIDAS Users Guide, volume A

**Note:** The total no. of entries + last entry no. are also written into integer keyword OUTPUTI(1,2).
See the help of the SET/ICAT command to find out how to make a catalog "active"

**Examples:** `SHOW/ICAT vinchuga`
Display the no. of entries + last entry no. in image catalog vinchuga.cat

## SHOW/KEYWORDS *core* 17-JUL-1989 KB

**Purpose:** Display contents of keyword data base.

**Syntax:** `SHOW/KEYWORDS [keyword]`

| | |
|---|---|
| `keyword` | optional keyword name;<br>If omitted, the internal layout of the complete keyword data base is shown, else the internal info of the given keyword is displayed |

**See also:** READ/KEY, WRITE/KEY, HELP/KEY, DELETE/KEY

**Note:** none

**Examples:** `SHOW/KEYWORDS IDIDEV`
Display the internal structure of the keyword IDIDEV

# SHOW/TABLE

## SHOW/TABLE *core* 17-OCT-1983 JDP

**Purpose:** Display table parameters

**Syntax:** `SHOW/TABLE table`

`table` name of table file

**Output:** The following information is stored in keywords :

OUTPUTI(1) - No. of columns in the table (2) - No. of rows in the table (3) - Number of the sorted column,( 0 if no sorted) (4) - Number of the column used as reference, (0 if SEQUENCE) (5) - Number of allocated rows (6) - Number of allocated columns (7) - Store value, as 0 (columnwise) / 1 (rowwise) (8) - Number of selected rows These informations are also stored in the descriptor TBLCONTR of the table.

**Note:** The text "Old transposed format" means that your table was created with the Old Midas Table File system. Any operation which will modify the table will convert it to the New Table File system (implemented in MIDAS since the 91MAY release). This new format IS NOT readable by MIDAS releases older than 91MAY. The command RETRO/TAB is available to convert the new format into the old one.

**See also:** READ/DESCR mytable.tbl *

**Examples:** `SHOW/TABLE mytable`

## SHOW/TCAT *core* 10-OCT-1988 KB

**Purpose:** Display the total no. of entries + last entry no. in a table catalog.

**Syntax:** `SHOW/TCAT [cat_name] [display_flag]`

`cat_name` name of table catalog;
defaulted to currently active table catalog

`display_flag` D(isplay) or N(odisplay), for displaying the information on the terminal or not;
defaulted to D

**See also:** READ/TCAT, SUBTRACT/TCAT, ADD/TCAT
Chapter 3, MIDAS Users Guide, Volume A

**Note:** The total no. of entries + last entry no. are also written into integer keyword OUTPUTI(1,2).
See the help of the SET/TCAT command to find out how to make a catalog "active"

**Examples:** `SHOW/TCAT vinchuga`
Display the no. of entries + last entry no. in table catalog vinchuga.cat

## SORT/FCAT *core* 24-JUN-1991 KB

**Purpose:** Sort a fit file catalog.

**Syntax:** `SORT/FCAT [cat_name]`

`cat_name` name of Fit File Catalog;
defaulted to the currently active Fit File Catalog

**See also:** SORT/TCAT, SORT/ICAT, SET/FCAT, CREATE/FCAT
READ/FCAT, ADD/FCAT, SUBTRACT/FCAT, SEARCH/FCAT

**Note:** Sorting is done according to the contents of the IDENT field in the catalog (i.e. usually the same as descr. IDENT).
The sorted catalog is first stored in a temporary file midtemp.cat, then the input catalog is deleted and, finally, midtemp.cat is renamed back to the original cat_name.
Therefore the input catalog should not be opened at the same time by e.g. another MIDAS session.

**Examples:** `SORT/FCAT`
Sort the currently active Fit File Catalog.

`SORT/FCAT zopilote`
Sort the fit file catalog 'zopilote.cat' .

## SORT/ICAT *core* 24-JUN-1991 KB

**Purpose:** Sort an image catalog.

**Syntax:** `SORT/ICAT [cat_name]`

`cat_name` name of Image Catalog;
defaulted to the currently active Image Catalog

**See also:** SORT/TCAT, SORT/FCAT, SET/ICAT, CREATE/ICAT
READ/ICAT, ADD/ICAT, SUBTRACT/ICAT, SEARCH/ICAT

**Note:** Sorting is done according to the contents of the IDENT field in the catalog (i.e. usually the same as descr. IDENT).
The sorted catalog is first stored in a temporary file midtemp.cat, then the input catalog is deleted and, finally, midtemp.cat is renamed back to the original cat_name.
Therefore the input catalog should not be opened at the same time by e.g. another MIDAS session.

**Examples:** `SORT/ICAT`
Sort the currently active Image Catalog.

`SORT/ICAT zopilote`
Sort the image catalog 'zopilote.cat' .

# SORT/TABLE

## SORT/TABLE *core* 12-Oct-1983 JDP + FO

**Purpose:** Sort table according to a combination of columns

**Syntax:** `SORT/TABLE table sort_keys`

| | |
|---|---|
| `table` | name of table file |
| `sort_keys` | sequence of columns to be used as sorting parameters, separated by commas; each column name may be followed by by (-) (a minus sign within brackets) for a descending sorting order. |

**Note:** none

**Examples:** `SORT/TABLE mytable :HD_NUMBER`
This would sort all the table columns in ascending of values stored in the column labelled :HD_NUMBER.

`SORT/TABLE mytable :ra,:dec(-)`
This would sort the table in ascending order of values stored in the column labelled :ra, and in case of identical :ra values, the order is the decreasing values of the :dec column.

## SORT/TCAT *core* 24-JUN-1991 KB

**Purpose:** Sort a table catalog.

**Syntax:** `SORT/TCAT [cat_name]`

| | |
|---|---|
| `cat_name` | name of Table Catalog;<br>defaulted to the currently active Table Catalog |

**See also:** SORT/ICAT, SORT/FCAT, SET/TCAT, CREATE/TCAT
READ/TCAT, ADD/TCAT, SUBTRACT/TCAT, SEARCH/TCAT

**Note:** Sorting is done according to the contents of the IDENT field in the catalog (i.e. usually the same as descr. IDENT).
The sorted catalog is first stored in a temporary file midtemp.cat, then the input catalog is deleted and, finally, midtemp.cat is renamed back to the original cat_name.
Therefore the input catalog should not be opened at the same time by e.g. another MIDAS session.

**Examples:** `SORT/TCAT`
Sort the currently active Table Catalog.

`SORT/TCAT zopilote`
Sort the table catalog 'zopilote.cat' .

## STATISTICS/IMAGE *core* 13-SEP-1995 KB

**Purpose:** Calculate statistics of a frame.

**Syntax:** `STATISTICS/IMAGE [frame] [area] [bins] [lo,hi_exc] [options] [outtab] [plotflg] [format]`

| | |
|---|---|
| `frame` | name of image frame; defaulted to displayed image |
| `area` | window within image to work on;<br>(a) subframe in the usual MIDAS form: [xs,ys:xe,ye], see Help of READ/IMAGE for more info about the syntax;<br>(b) complete frame, indicated by '+' (the plus sign);<br>(c) CURSOR for cursor defined windows, size may be adjusted interactively; or CURSOR,cmax for max. 'cmax' cursor inputs;<br>(d) ROW for row processing of 2-dim input frame;<br>(e) COLUMN for column processing of 2-dim input frame;<br>(f) PLANE for plane processing of 3-dim input frame<br>(g) table_name for table defined windows;<br>defaulted to (b); |
| `bins` | bin size for histogram calculation or no. of bins;<br>to distinguish between the two options, the character # is used for no. of bins;<br>defaulted to #256;<br>if the parameter lo,hi_exc is given, two excess bins are added; |
| `lo,hi_exc` | optional low and high excess values, i.e. end of lower excess bin and start of higher excess bin in the histogram. The resulting histogram is stored in the descriptor HISTOGRAM and can be converted to image or table format by the application procedure @a histogram (see HELP/APPLIC histogram); |
| `options` | 2 characters for level of statistics and level of display;<br>statistics level: F or G or R or X or S or H or M;<br>F(ull) => (1) - (4) is calculated, G like Full, but exact median,<br>R(educed) => (1) -> (3), X like Reduced, but also exact median,<br>S(hort) => (1) + (2), H(istogram) => (1) + (4),<br>M(inmax) => (1) - see the Note for details.<br>display level: F(ull), S(hort), X(short) or N(o) display;<br>defaulted to FF |
| `outtab` | optional output table (only for area option (c) -> (g)) with columns holding contents like desc. STATISTIC (cf. Note); |
| `plotflg` | P(lot) or N(oplot) for plotting the histogram (with some consequences for area option (c) -> (g));<br>defaulted to N |
| `format` | Format string used for displaying minimum, maximum, mean and std_dev of frame in Fortran notation;<br>defaulted to E15.6 |

**See also:** PLOT/HISTOGRAM, FIND/MINMAX, HELP/APPLIC histogram

# STATISTICS/IMAGE

**Note:** The parameters may also be referenced via
FRAME=, AREA=, BINSIZE=, EXCESS=, OPTION=, OUTTAB=, PLOT=, FORMAT=
Compute 1) minimum + maximum
2) mean value, stand. deviation
3) 3rd, 4th moment + intensity (sum of pixels)
4) histogram + median + first mode + mode of data values.
The following descriptors are filled for area option (a) and (b):
WINDOW_FROM, WINDOW_TO, STATISTIC and if options(1) = F or H, also HIST_BINS and HISTOGRAM.
Start and end frame pixels of the area are saved in integer descr. WINDOW_FROM and WINDOW_TO.
Real descr STATISTIC is filled depending upon options(1) as follows:
Minimum(1), maximum(2), mean value(3), standard deviation(4), 3rd moment(5), 4th moment(6), intensity(7), median(8), first mode(9), no.of bins(10), binsize(11), mode(12).
By default, the median value is determined via the histogram of the image, in order to speed up the STATISTICS command. This works pretty well if there are no "runaway" data, like e.g. saturated columns in a CCD frame which are far outside the rest of the data intensity range. But, note, by using well chosen excess bins, also in that case we would get a good approximation to the median.
With the statistics level (1. char. of parameter 'options') set to G or X you enforce the precise calculation of the median value, which means that the complete image had to be sorted.
The difference in execution time for the F or G statistics level is negligible for small frames, i.e. images with up to 512*512 pixels. For larger images, however, the exact calculation of the median takes significantly longer: approx. 4 times for a 1024*1024 image, and about 7 times for a 2048*2048 image.
For options(1) = F/G: elements (1) - (12) of STATISTIC are written,
R: (1) - (7), X: (1) - (8), S: (1) - (4), M/H: (1) - (2).
The number of bins + binsize is stored in the real descr. HIST_BINS in element (1,2), lo,hi_excess are stored in (3,4) and the no. of excess bins (= 0 or 2) is put in HIST_BINS(5).
Integer descr HISTOGRAM is filled with histogram:
HISTOGRAM(1) = number of values in first bin, which is the lower excess bin, if lo,hi_excess are given
HISTOGRAM(2) = number of values in second bin, ...,
HISTOGRAM(n) = number of values in last bin, which may be the high excess bin, n = HIST_BINS(1)
Real keyword OUTPUTR is filled with same contents as STATISTIC.
Integer keyword OUTPUTI will hold total no. of pixels and pixel numbers of min. and max. coordinates, i.e. (1 + NAXIS*2) values.
If the complete frame was processed, descriptor LHCUTS(3,4) will also be set to the minimum and maximum value.
If an output table name is specified, the statistical quantities are stored in columns which are labeled
:MIN, :MAX, :MEAN, :STD, :MOM3, :MOM4, :TOT_INTENS, :MEDIAN, :SMAL_MODE, :NO_BINS, :BIN_SIZE and :MODE.
If the plotflag is set to 'P', statistic level must be = F, G or H, otherwise there is no histogram! In that case, also for area options (c) -> (g) the descr. HIST_BINS and HISTOGRAM are always updated.

**Examples:** `STATIST/IMAGE toro [335,405:3333,425] 0.5`
Work in specified window of image frame 'toro.bdf'.

`STATIST/IMAGE ? CURSOR`
Work on displayed image, choose windows via cursor.

`STATIST/IMAGE becerra BINSIZE=1.0 EXCESS=0.22,9.8`
Do statistics on image frame 'becerra.bdf', specify a binsize of 1.0 - all pixels with intensity up to 0.22 are collected in the low excess bin, all pixels with intensity higher than 9.8 are collected in the high excess bin.

`STATIST/IMAGE vaca PLOT=P`
Statistics on image frame 'vaca.bdf', no display of results, but plot the histogram.

`STATIST/IMAGE vaca option=GF plot=P`
As above but compute the exact value of the median.

`STATIST/IMAGE toro ROW OPTION=RN OUTTAB=mytab`
Reduced statistics on all rows of image frame 'toro.bdf'. Display nothing and write results to table file 'mytab.tbl'.

`STATIST/IMAGE chiva cursor option=ss format=f5.2`
Use cursor rectangle to define areas for calculating short statistics (and short display) and use format 'F5.2' for output. Frame 'chiva.bdf' is loaded in the display.

## STATISTICS/TABLE *core* 08-OCT-1985 JDP

**Purpose:** Calculate statistics of a table column Compute 1) total number of entries and selected entries 2) minimum + maximum 3) mean value + standard deviation

**Syntax:** `STATISTICS/TABLE table column`

`column` column reference by number or by label

**See also:** PLOT/HIST, COMPUTE/HIST, READ/HIST, PRINT/HIST

**Note:** Real keyword OUTPUTR is filled with: Minimum(1), maximum(2), mean value(3), standard deviation(4) Integer keyword OUTPUTI is filled with: Column number(1), number of selected non-null values (2)

**Examples:** `STATISTICS/TABLE mytable :VALUE`

# STORE/FRAME

STORE/FRAME *core* 16-JAN-1990 KB

**Purpose:** Store frame or entries of a catalog into keyword

**Syntax:** `STORE/FRAME key frame [indx] [exit_label]`

| | |
|---|---|
| `key` | name of keyword where frame or catalog entry will be stored in |
| `frame` | a) name of frame (syntax = nnnnn);<br>b) name of catalog (syntax = nnnnn.cat) |
| `indx` | index into local keyword CATAL, only used with option b);<br>defaulted to 1, i.e. use CATAL(1) |
| `exit_label` | label to jump to on termination of catalog processing, only used with option b) |

**See also:** WRITE/KEY, all catalog commands

**Note:** for a)
the command is the same as the commmand: WRITE/KEYWORD key frame
for b)
the command has to be imbedded into a loop, e.g.:
(1) DEFINE/LOCAL CATAL/I/1/1 0
(2) CAT_LOOP:
(3) STORE/FRAME IN_A astro.cat 1 FINITO
(4) .... commands using frames stored in key IN_A ....
(5) GOTO CAT_LOOP
(6) FINITO:
(7) .... any other commands ....
The local key CATAL is defined in line (1), it must be initialized to zero to indicate the start of the loop. In line (3) the next entry in the specified catalog will be stored into IN_A. This entry no. will be put into CATAL(1) and will be set to -1 upon termination of the loop.
The type of the catalog will determine if images, tables or fit files are stored in the keyword.

**Examples:** `see above`

## SUBTRACT/ACAT *core* 20-JUN-1991 KB

**Purpose:** Remove entries from an ASCII file catalog.

**Syntax:** `SUBTRACT/ACAT [cat_name] [frame_list]`

| | |
|---|---|
| `cat_name` | name of ASCII file catalog;<br>defaulted to currently active ASCII file catalog; if there is an ASCII catalog enabled (cf. SET/ACAT), 'cat_name' is defaulted to this "active" ASCII catalog |
| `frame_list` | list of files to be subtracted; may be:<br>file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*);<br>or numbers referring to the names (e.g. #7,#1). |

**Note:** The different options for the frame_list may not be mixed!

**See also:** ADD/ACAT, READ/ACAT, CREATE/ACAT

**Examples:**

`SUBTRACT/ACAT dec88 bar.chart`
Remove the entry for file bar.chart from catalog dec88.cat.

`SUBT/ACAT dec88 zap*`
Remove entries for all files where the names begin with the string "zap" from catalog dec88.cat.

`SUBT/ACAT dec88 nov89.cat`
Remove entries for all files which are in the ASCII catalog nov89.cat from catalog dec88.cat.

`SUBT/ACAT dec88 #1,#6`
Remove entries no. 1 and no. 6 from catalog dec88.cat.

# SUBTRACT/FCAT

## SUBTRACT/FCAT *core* 20-JUN-1991 KB

**Purpose:** Remove entries from a fit file catalog.

**Syntax:** `SUBTRACT/FCAT [cat_name] [frame_list]`

| | |
|---|---|
| `cat_name` | name of fit file catalog;<br>if there is a fit file catalog enabled (cf. SET/FCAT), defaulted to currently active fit file catalog |
| `frame_list` | list of fit files to be subtracted,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.fit);<br>or numbers referring to the names (e.g. #7,#1) |

**Note:** The different options for the frame_list may not be mixed!

**See also:** SET/FCAT, CREATE/FCAT, CLEAR/FCAT, READ/FCAT, ADD/FCAT
Chapter 3 of the MIDAS Users Guide, Volume A

**Examples:** `SUBTRACT/FCAT dec88 func01`
Remove entry for fit file func01.fit from catalog dec88.cat.

`SUBT/FCAT dec88 func*`
Remove entries for all fit files where the names begin with the string "func" from catalog dec88.cat.

`SUBT/FCAT dec88 nov89.cat`
Remove entries for all files which are in the fit file catalog nov89.cat from fit file catalog dec88.cat.

`SUBT/FCAT dec88 #1,#6`
Remove entries no. 1 and no. 6 from catalog dec88.cat.

SUBTRACT/ICAT *core* 20-JUN-1991 KB

**Purpose:** Remove entries from an image catalog.

**Syntax:** `SUBTRACT/ICAT [cat_name] [frame_list]`

| | |
|---|---|
| `cat_name` | name of image catalog;<br>if there is an image catalog enabled (cf. SET/ICAT), 'cat_name' is defaulted to this "active" image catalog |
| `frame_list` | list of frames to be subtracted,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.bdf);<br>or numbers referring to the names (e.g. #7,#1) |

**Note:** The different options for the frame_list may not be mixed!

**See also:** SET/ICAT, CREATE/ICAT, CLEAR/ICAT, READ/ICAT, ADD/ICAT
Chapter 3 of the MIDAS Users Guide, Volume A

**Examples:** `SUBTRACT/ICAT dec88 galax001`
Remove the entry for image frame galax001.bdf from catalog dec88.cat

`SUBT/ICAT dec88 gal*`
Remove entries for all image frames where the names begin with the string "gal" from catalog dec88.cat.

`SUBT/ICAT dec88 nov89.cat`
Remove entries for all frames which are in the image catalog nov89.cat from catalog dec88.cat.

`SUBT/ICAT dec88 #1,#6`
Remove entries no. 1 and no. 6 from catalog dec88.cat.

# SUBTRACT/TCAT

## SUBTRACT/TCAT *core* 20-JUN-1991 KB

**Purpose:** Remove entries from an table catalog.

**Syntax:** `SUBTRACT/TCAT [cat_name] [frame_list]`

| | |
|---|---|
| `cat_name` | name of table catalog;<br>defaulted to currently active table catalog |
| `frame_list` | list of table files to be subtracted,<br>maybe file specifications, separated by a comma (no spaces!);<br>or a single catalog name;<br>or wildcard specifications (e.g. a3*,n*.tbl);<br>or numbers referring to the names (e.g. #7,#1) |

**Note:** The different options for the frame_list may not be mixed!

**See also:** SET/TCAT, CREATE/TCAT, CLEAR/TCAT, READ/TCAT, ADD/TCAT

**Examples:** `SUBTRACT/TCAT dec89 coords`
Remove the entry for table file coords.tbl from catalog dec89.cat.

`SUBT/TCAT dec88 coo*`
Remove entries for all tables where the names begin with the string "coo" from catalog dec88.cat.

`SUBT/TCAT dec88 nov89.cat`
Remove entries for all tables which are in the table catalog nov89.cat from catalog dec88.cat.

`SUBT/TCAT dec88 #1,#6`
Remove entries no. 1 and no. 6 from catalog dec88.cat.

## SYNCHRONIZE/MIDAS *core* 20-OCT-1993 KB

**Purpose:** Write keyfile and logfile to disk.

**Syntax:** `SYNCHRONIZE/MIDAS`

**See also:** BYE

**Note:** The keyword file and the log file are only written to disk when an application command, e.g. LOAD/IMAGE is executed.
So, if your application (GUI) needs the most recent updates to the keyword data base you must execute SYNCHRONIZE/MIDAS first.

**Examples:** `SYNC/MIDAS`

TRANSLATE/SHOW *core* 11-APR-1994 KB

**Purpose:** Translate Midas procedure and display resulting code.

**Syntax:** `TRANSLATE/SHOW proc option`

| | |
|---|---|
| `proc` | name of MIDAS command procedure, file type defaulted to '.prg' |
| `option` | N (normal) or X (extended);<br>for option = 'X' all Midas commands in the procedure are checked for completeness, i.e if commands and qualifiers are fully specified and not abbreviated;<br>only lines with offending commands are displayed<br>if set to 'X,silent' individual command lines are not displayed, just a final message appears, if some commands/qualifiers are abbreviated;<br>for option = 'N' the procedure is translated to the internal Midas code and all lines are displayed on the terminal;<br>defaulted to N |

**See also:** SHOW/CODE, @@, DEBUG/PROCEDURE, system procedure progcheck.prg

**Note:** To see the translation for a Midas system procedure use
TRANSLATE/SHOW MID_PROC:procedure .
For option = 'X' the keyword MID$INFO(1) is set to the no. of offending command lines.

**Examples:** `TRANSLA/SHOW lobo`
Translate procedure 'lobo.prg' (in current directory) and display its code.

`TRANSLA/SHOW lobo x`
Translate procedure 'lobo.prg' (in current directory), check all Midas commands in that procedure for completeness and display only the lines with incomplete commands or qualifiers.

`TRANSLA/SHOW MID_WORK:chiva`
Translate procedure 'chiva.prg' (in MID_WORK directory) and display its code.

# TRANSPOSE/CUBE

## TRANSPOSE/CUBE *core* 21-SEP-1995 KB

**Purpose:** Rearrange the planes of a cube.

**Syntax:** `TRANSPOSE/CUBE inframe [outframe] [plane_spec]`

| | |
|---|---|
| `inframe` | input image (of 3 dimensions) |
| `outframe` | output image;<br>if omitted, the input image will be rearranged in place |
| `plane_spec` | plane specification (2 chars.), controls which planes of inframe will become the planes in xy-direction of outframe;<br>if set to ZY the zy_planes of inframe become the new xy_planes;<br>if set to XZ the xz_planes become the new xy_planes;<br>defaulted to ZY |

**See also:** COMPUTE/XYPLANE, COMPUTE/XZPLANE, COMPUTE/ZYPLANE
TRANSPOSE/IMAGE, FLIP/IMAGE

**Note:** The xy_plane of a cube is formed by the x- and y-axis and has NPIX(1) pixels per row and NPIX(2) pixels per column.
The zy_plane of a cube is formed by the z- and y-axis and has NPIX(3) pixels per row and NPIX(2) pixels per column.
The xz_plane of a cube is formed by the x- and z-axis and has NPIX(1) pixels per row and NPIX(3) pixels per column.
The planes of the input cube are rearranged in such a way as to keep their respective start and step values unchanged. That means, that the TRANSPOSE command does not just provide different views at the cube as if one turned around a brick.

**Examples:** `TRANSPOSE/CUBE chico ? xz`
Rearrange the planes of 'chico.bdf' in such a way, that the xz-coords. become the new xy-coords. The first new xy-plane is the xz-plane which was located at the start y-coordinate of 'chico.bdf' and the last new xy-plane is the xz-plane located at the end y-coord. of the frame. The contents of 'chico.bdf' are overwritten.

`TRANSPOSE/CUBE largo otro`
Rearrange the planes of 'largo.bdf' in such a way, that the zy-coords. become the xy-coords. of 'otro.bdf'. The first xy-plane of 'otro.bdf' is the zy-plane of 'largo.bdf' which was located at the start x-coordinate of 'largo.bdf' and the last xy-plane of 'otro.bdf' is the zy-plane located at the last x-coord. of 'largo.bdf'. The x-coords. of 'largo.bdf' become the z-coords of 'otro.bdf'.

## TRANSPOSE/IMAGE *core* 13-JAN-1992 KB

**Purpose:** Transpose an image in the sense of matrix algebra.

**Syntax:** `TRANSPOSE/IMAGE inima outima [diagonal]`

| | |
|---|---|
| `inima` | name of input image |
| `outima` | name of output image |
| `diagonal` | MAJOR or MINOR to indicate major/minor diagonal; defaulted to MAJOR |

**See also:** FLIP/IMAGE, TRANSPOSE/CUBE

**Note:** Transposing around the major axis is equivalent to flipping in y and clockwise rotation around 90 degrees.
Transposing around the minor axis is equivalent to flipping in x and clockwise rotation around 90 degrees (rows change to columns).
The start and step values are updated accordingly.

**Examples:** `TRANSPOSE/IMAGE becerra vaca`
Transpose image 'becerra.bdf' around major diagonal and put result into 'vaca.bdf'.

`TRANSPOSE/IMAGE becerra toro minor`
Transpose image 'becerra.bdf' around minor diagonal and put result into 'toro.bdf'.

## TUTORI/EXTR *core* 25-APR-1991 KB

**Purpose:** Demonstrate some of the different EXTRACT commands.

**Syntax:** `TUTORIAL/EXTRACT`

**See also:** All EXTRACT/... commands

**Note:** None

**Examples:** `TUTORIAL/EXTRACT`
run the tutorial

## TUTORI/FILT *core* 03-JUL-1990 KB

**Purpose:** Explain the usage of filters.

**Syntax:** `TUTORIAL/FILTER`

See also: TUTORIAL/..., FILTER/..., FFT/...

**Note:** None

**Examples:** `TUTORIAL/FILTER`
Run the tutorial.

# TUTORI/GRAPHICS

## TUTORI/GRAPHICS *core* 03-JUL-1990 RHW

**Purpose:** Explain the use of the graphics package

**Syntax:** `TUTORIAL/GRAPHICS option`

| | |
|---|---|
| `option` | GENERAL to show the general features in the plot package<br>= AXES to show how to make a multi-axes plot layout<br>= TABLE for table related plot commands<br>= TBL3 for 3-D table related plot commands<br>= 1DIM for spectral related plot commands<br>= 2DIM for two dimensional image related plot commands<br>defaulted to GENERAL |

**Note:** none

**See also:** none

**Examples:** `TUTORIAL/GRAPHICS table`
Show the different plot commands related to MIDAS tables.

## TUTORI/HELP *core* 03-JUL-1990 KB

**Purpose:** Explain usage of the HELP command.

**Syntax:** `TUTORIAL/HELP`

**See also:** All HELP/... commands

**Note:** None

**Examples:** `TUTORIAL/HELP`
run the tutorial

## TUTORI/ITT *core* 03-JUL-1990 KB

**Purpose:** Explain the usage of ITT's.

**Syntax:** `TUTORIAL/ITT [plotflag]`

| | |
|---|---|
| `plotflag` | Plot or NoPlot, for plotting the ITT or not;<br>defaulted to Plot |

**See also:** TUTORIAL/LUT

**Note:** None

**Examples:** `TUTORIAL/ITT Plot`
run the tutorial and use also the graphics display

## TUTORI/LUT *core* 03-JUL-1990 KB

**Purpose:** Show some standard LUT's and related MIDAS commands.

**Syntax:** `TUTORIAL/LUT [plotflag]`

| | |
|---|---|
| `plotflag` | Plot or NoPlot, for plotting the red, green and blue component of the LUT or not; defaulted to Plot |

**See also:** TUTORIAL/ITT

**Note:** None

**Examples:** `TUTORIAL/LUT`
run the tutorial but omit the graphical display of the LUTs

## TUTORI/SPLI *core* 03-JUL-1990 KB

**Purpose:** Explain the usage of split screen.

**Syntax:** `TUTORIAL/SPLIT`

**See also:** TUTORIAL/..., SET/SPLIT, CLEAR/SPLIT

**Note:** The commands SET/SPLIT and CLEAR/SPLIT are only supported on DeAnza image processors.

**Examples:** `TUTORIAL/SPLIT`
Run the tutorial.

## TUTORI/TABL *core* 03-JUL-1990 JDP,MP

**Purpose:** Explain usage of tables.

**Syntax:** `TUTORIAL/TABLE`

**Note:** none

**Examples:** `TUTORIAL/TABLE`
run the tutorial

| $ | *core* | 15-JUL-1992 KB |
|---|---|---|

**Purpose:** Execute a host (operating system) command. Currently supported host systems are VAX/VMS and all major UNIX systems.

**Syntax:** `$ host-comnd`

`host-comnd` a UNIX (Bourne-shell) or VMS (DCL) command to be executed

**See also:** SET/MIDAS_SYSTEM
Chapter 3 of the MIDAS Users Guide, Vol. A

no Note: Before passing the command line to the host system all substitutions of MIDAS are performed. If you want to pass the line "as it is", use: $$ host-comnd .
But note, that the $$ command only works interactively...

**Examples:** `$ date`
Get the current time in UNIX.

`$ show time`
Get the current time in VMS.

`$$ cp /rio/grande/elpaso.bdf .`
In Unix, copy the file elpaso.bdf in directory /rio/grande to current directory with same name. If only a single '$' would be used the name of the new file would depend upon the contents of the preceding line, a name like '?' is quite likely...

`$$ delete [rio.grande]elpaso.bdf;*`
In VMS, delete the file elpaso.bdf in directory [rio.grande]. Again, a single '$' would lead to a VMS error message, since the line would be split up into 2 MIDAS commands...

## VIEW/IMAGE *core* 14-DEC-1994 KB

**Purpose:** View an image with a "looking glass" and get statistics.

**Syntax:** `VIEW/IMAGE [frame] [out_tab] [plot_option] [g,zhardcopy]`

| | |
|---|---|
| `frame` | name of image, the image will be loaded with the cuts set to mean-3*sigma, mean+3*sigma<br>if omitted, the currently displayed image is used |
| `out_tab` | name of optional output table for coordinates of visited regions |
| `plot_option` | P or N, for P(lotting) or N(oPlotting), defaulted to P; |
| `g,zhardcopy` | names of plotters for hardcopies of graphics window (initiated via key 'p') and zoom window (initiated via key 'q');<br>defaulted to LASER,LASER |

**See also:** STATISTICS/IMAGE, GET/CURSOR, CENTER/GAUSS, MAGNITUDE/RECTANGLE EXTRACT/CURSOR, EXTRACT/RTRACE, LOAD/IMAGE, CUTS/IMAGE

**Note:** This command is only implemented for XWindow stations!
If the zoom window does not exist yet, it will be created with half the size in x and y of main display window. Also a graphics window will be opened if plot_option = P.
Use the cursor to select the center of subimages to be viewed. These subimages are zoomed according to the current zoom factor and displayed in the zoom_window.
This command has lots of different options which are activated via different keys on the keyboard. But note, that the cursor has to remain in the display window at all times, even if you type on the keyboard.
The available options are:
h = get this help, z = zoom up, x = zoom down c = load colour LUT, b = load b+w LUT, l = modify LUT via arrow keys i = load ITT, j = clear ITT, k = modify ITT via arrow keys u = toggle looking glass mode (zoom on the fly) m = modify cuts and redisplay subimage or full image t = toggle plot options, s = toggle cut options in zoom window v = toggle statistics/magnitude option a = modify radius for magnitude, nomansland, background p or q = make hardcopy of graphics or zoom window, e = extract subimage if display memory > display window, use the arrow keys to scroll
The optional output table will have columns with the standard labels :XSTART, :YSTART, :XEND and :YEND.
If the plotting option is set to row/column-trace mode, not the 2-dim region of interest is extracted with the 'e' command, but the row or column at the center (the one which is plotted).

**Examples:** `VIEW/IMAGE cielo`
Load image 'cielo.bdf' with low, high cut values set to mean-3*sigma and mean+3*sigma.
Use the default size (or size of existing auxiliary window) for the zoom window.

`VIEW/IMAGE cielo ? ? ps4ipg1`
As above, but send all eventual hardcopies to printer with the name 'ps4ipg1'.

`VIEW/IMAGE ? luna`
Work on image which is currently loaded in the display, write the coordinates of all regions which were visited into output table 'luna.tbl'.

`VIEW/IMAGE estrella.mt`
The FITS file 'estrella.mt' is loaded into the display window and worked on like any binary Midas frame.

# WAIT/BACK_MIDAS

## WAIT/BACK_MIDAS *core* 12-MAR-1990 KB

**Purpose:** Wait until command in background MIDAS terminates.

**Syntax:** `WAIT/BACK_MIDAS [unit]`

| | |
|---|---|
| `unit` | unit of background Midas;<br>if unit is omitted, we wait on outstanding commands sent to each background Midas |

**See also:** CONNECT/BACK_MIDAS, [BackgroundMidas]

**Note:** Depending upon the CONNECT/BACK command, we wait or don't wait for the commands executed in a background Midas.
This command may be used to wait for commands which have been sent to a background Midas in 'no_wait' mode.

**Examples:** `WAIT/BACK zk`
Wait until current command in background Midas with unit zk is terminated.

`WAIT/BACK`
Wait until all commands in any background Midas are terminated.

## WAIT/SECS *core* 23-DEC-1993 KB

**Purpose:** Suspend MIDAS monitor for a number of seconds.

**Syntax:** `WAIT/SECS [no_of_secs]`

| | |
|---|---|
| `no_of_secs` | no of seconds to wait; defaulted to 1 sec |

**See also:** WAIT/BACK_MIDAS

**Note:** If 'no_of_seconds' is < 1 then the WAIT/SECS command works like a NOOP (No Operation) command.

**Examples:** `WAIT/SECS 12`
Wait 12 seconds before continuing with next command.

`WAIT/SECS 0`
Do not wait but continue immediately - this may be useful if you need a NOOP command in a branch of an IF statement.

WRITE/COMMANDS *core* 09-OCT-1991 KB

**Purpose:** Save commands from command buffer + write them into a procedure file

**Syntax:** `WRITE/COMMANDS [procnam] [par1] [par2] ... [par8]`

`procnam` name of procedure file where the commands should be stored;
if omitted, the commands are stored in the file 'midtempXY.prg', where XY is the MIDAS unit, in the MIDAS working directory specified by the variable MID_WORK

`par1 ... par8` the parameters for the procedure,
each command line in the command buffer is scanned for an occurrence of par1, par2, ...
if a complete (case sensitive + no abbreviations!) match is found, that substring will be replaced by Pi

**See also:** READ/COMMANDS, CLEAR/BUFFER, SET/BUFFER
Chapter 3, MIDAS Users Guide, Volume A

**Note:** This command creates a new procedure quickly from the commands you just used interactively and can be edited later on with a normal text editor.

**Examples:** `WRITE/COMM delicias`
If we assume the following MIDAS command buffer:
1 READ/KEY MODE
2 WRITE/KEY INPUTC BRAVO
3 READ/KEY INPUTC
we create the file delicias.prg, containing:
READ/KEY MODE
WRITE/KEY INPUTC BRAVO
READ/KEY INPUTC
in the current directory.

`WRITE/COMM`
Will create file midtemp34.prg (assuming you use MIDAS unit 34) in the directory specified via MID_WORK with the same contents.

`WRITE/COMMANDS torreon ? BRAVO`
Will create procedure torreon.prg in current directory with:
READ/KEY MODE
WRITE/KEY INPUTC P2
READ/KEY INPUTC

# WRITE/DESCR

WRITE/DESCR *core* 16-MAR-1993 KB

**Purpose:** Store values into a descriptor.

**Syntax:** `WRITE/DESCR frame descr data [flg]`

| | |
|---|---|
| `frame` | name of data file |
| `descr` | complete descriptor specification, name/type/felem/nval, e.g. NPIX/I/1/2<br>or only descriptor name, e.g. NPIX, if the descriptor exists already.<br>'nval' data values will be written into the descriptor 'name' of given type, beginning at element 'felem'. |
| `data` | data values, separated by a comma |
| `flg` | optional flag, if set to ALL, the descriptor will be filled completely with the value given as data, so you set all descriptor elements to that value |

**See also:** READ/DESCR, SHOW/DESCR, PRINT/DESCR, DELETE/DESCR, COPY/DD
WRITE/DHELP, @a dscedit
chapter 3 of MIDAS Users guide, volume A

**Note:** Valid types are I for integer, R for real, D for double precision, C for character descriptors and C*nnn for character array descriptors.
Descriptors are extended automatically (like a file is extended if needed when it is edited) by writing more elements or by starting at an element larger than the no. of elements defined in the descriptor creation.
Beware, you can create holes like that...
To edit an existing descriptor use the application procedure 'dscedit.prg', use "HELP/APPLIC dscedit" to get details.

**Examples:** `WRITE/DESCR dinosaur RR/R/1/5 1.,2.,3.,4.,5.`
Set real descriptor elements RR(1), ..., RR(5) of frame 'dinosaur.bdf' to 1.,2.,3.,4.,5. If the descriptor RR had been created before with 3 elements only, it would be extended automatically.

`WRITE/DESCR fossil ck/c/1/20 "abcdefghij 123456789"`
Define character descriptor 'ck' as a "flat" string, like
"CHARACTER CK*20" in FORTRAN or "char ck[20]" in C,
and fill it with the 20 char. string 'abcdefghij 123456789'. Note, that the string must be enclosed in double quotes ("), because it contains a blank (which is the delimiter in MIDAS).

`WRITE/DESCR fossil cmore/c*5/1/10 abcde all`
Define character descriptor 'cmore' as a character array, like
"CHARACTER CMORE(10)*5" in FORTRAN or "char cmore[10][5]" in C,
and fill all its 10 elements (which are strings of 5 chars.) with the string 'abcde'.

`WRITE/DESCR hippo.tbl ir/i/1/5 55 all`
Set the first 5 elements of integer descriptor IR of table file 'hippo.tbl' to 55.

## WRITE/DHELP *core* 25-MAY-1993 KB

**Purpose:** Store help-text/comments for an existing descriptor.

**Syntax:** `WRITE/DHELP frame descr text`

| | |
|---|---|
| `frame` | name of data file |
| `descr` | descriptor name |
| `text` | help text, has to be enclosed in double quotes (") |

**See also:** READ/DESCR, SHOW/DESCR, WRITE/DESCR, DELETE/DESCR, COPY/DD
chapter 3 of MIDAS Users guide, volume A

**Note:** The text is extended automatically if different WRITE/DHELP commands are done for the same descriptor, but not resized (shrunk) if the new text is shorter than the previous one.

**Examples:** `WRITE/DHELP dinosaur xyztime "exposure time for instrument xyz"`
Store the comment for descr. 'xyztime' of frame 'dinosaur.bdf'. The descriptor 'xyztime' must exist already.

## WRITE/FILE *core* 23-SEP-1992 KB

**Purpose:** Write into an ASCII file which was opened before via the OPEN/FILE command.

**Syntax:** `WRITE/FILE file_id charbuf`

| | |
|---|---|
| `file_id` | file id which was returned by a previous OPEN/FILE command |
| `charbuf` | character buffer to be written into file |

**See also:** OPEN/FILE, CLOSE/FILE, READ/FILE

**Note:** Everything in the command line after the 'file_id' is written to the file including spaces. If you just want to write a blank line, the spaces have to be enclosed by double quotes ("...").
The actual no. of characters written into the file is stored as the 2nd element of the integer keyword specified in the related OPEN/FILE command.

**Examples:** `WRITE/FILE 8 Steglitz`
Write the string 'Steglitz' into ASCII file with file_id 8.

`WRITE/FILE fctr(1) Lankwitz Lichterfelde`
Write string 'Lankwitz Lichterfelde' into file with the file_id which was stored in integer keyword fctr(1) in a previous OPEN/FILE command.

`WRITE/FILE 7 p4(11:20)`
Write the contents of parameter P4 (beginning at element 11) into ASCII file with file_id 7.

`WRITE/FILE 7 " "`
Write a blank line into ASCII file with file_id 7.

# WRITE/IMAGE

## WRITE/IMAGE *core* 08-OCT-1991 KB

**Purpose:** Store values into image pixels.

**Syntax:** `WRITE/IMAGE frame_specs [pixel_specs] data [flg]`

| | |
|---|---|
| `frame_specs` | name of image data frame;<br>or CURSOR if cursor rectangle is used to define the region of the displayed frame which will be updated;<br>or CURSOR,1 if only one cursor is used for writing single pixels of the displayed frame |
| `pixel_specs` | string defining the frame interval to be updated;<br>(a) 'xs,noval', 'xs,ys,noval' or 'xs,ys,zs,noval' defining the start coords in x,y,z (according to the dimensions of the frame) and 'noval' the number of values to be written;<br>(b) '[xs:xe]', '[xs,ys:xe,ye]' or '[xs,ys,zs:xe,ye,ze]', noval will then be determined from size of interval<br>(c) 'tbname,TABLE' where the table 'tbname' must contain the columns :XSTART, :YSTART, :XEND, :YEND and :VALUE1 defining the windows in the frame which are set to the constant in :VALUE1;<br>defaulted to: '<,20', '<,<,20' or '<,<,<,20' for 1-, 2- or 3-dim frame, i.e. write 20 values at beginning of frame.<br>With the CURSOR option the parameter 'pixel_specs' has to be omitted! |
| `data` | data values separated by commas, will be interpreted as floating point data |
| `flg` | optional flag, if set to ALL, all pixels defined by the parameter 'pixel_specs' above (or cursor area) are set to the first value given in the parameter 'data' |

**See also:** READ/IMAGE, PRINT/IMAGE, REPLACE/IMAGE, COMPUTE/IMAGE, MODIFY/PIXEL, MODIFY/AREA

**Note:** For the MIDAS standard of specifying coordinates see the help of READ/IMAGE.
If no. of values given in 3rd parameter is less than 'noval' of parameter 'pixel_spec' only that many values are written.
For 'frame_specs' = CURSOR or for option (c) above, a single value is written to all pixels in the specified interval, i.e. the parameter 'flg' is not taken into account.
The columns required with the table option are exactly the ones of a table created via GET/CURSOR (with cursor-rectangle option).

**Examples:** `WRITE/IMAGE becerra <,<,4 1.0,2.0,3.0,4.0`
Write values 1.0,2.0,3.0,4.0 into 2-dim image 'becerra.bdf', starting at 1st pixel in x and y.

`WRITE/IMAGE becerra [@1,@1:@4,@1] 1.0,2.0,3.0,4.0`
Exactly the same command as above, with just another syntax for pixel specifications.

`WRITE/IMAGE vaca @20,@100,20 2.3 ALL`
Starting with pixel no. 20 in line no. 100, set 20 pixels of frame 'vaca.bdf' to the constant 2.3

`WRITE/IMAGE cursor 22.2`
Use the cursor rectangle to define interactively the region where all pixels of the curently displayed frame are set to 22.2

`WRITE/IMAGE perro gato,table`
Use the table 'gato.tbl' to specify intervals in the frame 'perro.bdf' and fill all pixels in these intervals with the value stored in column :VALUE1 of table 'gato.tbl'.

WRITE/KEYWORD *core* 16-MAR-1993 KB

**Purpose:** Store values into a keyword.

**Syntax:** `WRITE/KEYWORD key data [flg]`

| | |
|---|---|
| `key` | complete keyword specification;<br>either name/type/first_elem/noval, e.g. INPUTI/I/1/2<br>or only keyword name, e.g. INPUTI, if the keyword exists already;<br>as many as 'noval' data values will be written into the keyword 'name' of given type, beginning at the first element (which is 1 if just keyword name is given). |
| `data` | data values, separated by a comma |
| `flg` | optional flag, if set to ALL, the keyword will be filled completely with the value given as data, so you set all key elements to that value |

**See also:** DEFINE/LOCAL_KEYWORD, READ/KEY, SHOW/KEY, HELP/KEY, COPY/KEY, DELETE/KEYWORD
chapter 3 of MIDAS Users guide, volume A

**Note:** Valid types are I for integer, R for real, D for double precision, C for character keywords and C*nnn for character array keywords.
Contrary to descriptors, keywords cannot be extended once they have been defined with a given size.

**Examples:** `WRITE/KEY RR/R/1/5 1.,2.,3.,4.,5.`
Set real keyword elements RR(1), ..., RR(5) to 1.,2.,3.,4.,5.

`WRITE/KEY lola/I/1/5 22 ALL`
Set all elements of integer keyword lola to 22.

`WRITE/KEY petrita/c/6/5 zyxwvutsrq`
Change the contents of character keyword petrita(6:10) to the string 'zyxwv', so if petrita contained originally the string 'whatastupidexample', it will be changed to 'whatazyxwvdexample' (the additional characters are ignored).

`WRITE/KEYWORD clonk/c*5/1/10 abcde all`
Define character keyword 'clonk' as a character array, like
"CHARACTER CLONK(10)*5" in FORTRAN or "char clonk[10][5]" in C,
and fill all its 10 elements (which are strings of 5 chars.) with the string 'abcde'.

# WRITE/OUT



**Purpose:** Display text on terminal.

**Syntax:** `WRITE/OUT text_spec [section] [label]`

| | |
|---|---|
| `text_spec` | text string to be displayed (a) or name of text file (b)<br>(a) if the text string contains spaces it should be enclosed in double quotes (")<br>(b) complete specification of textfile, the type must be ".txt" and be appended to the name |
| `section` | char. string serving as section in textfile, only used in connection with text file (b) |
| `label` | char. string which is appended to section, to mark the beginning of the paragraph you want to display, only used in connection with text file (b) |

**See also:** WRITE/ERROR

**Note:** If you use WRITE/OUT with the text file option, then:
1) no variable substitution will be done in the text lines.
2) If section and label are omitted, the complete file will be displayed on the terminal (and stored into the logfile).

**Examples:** `WRITE/OUT I am here to look at you and your pretty pictures.`
This command line would result in the following text output:
I am here to look at you and yourprettypictures.
However,
WRITE/OUT "I am here to look at you and your pretty pictures." is correct and yields the desired output.

`WRITE/OUT display.txt`
displays the contents of ASCII file display.txt
assume, display.txt contains the following lines:
aaaaaaaaaaaaaaaa
Para_1
we all like weissbier
Paraxcf
bbbbbbbbbbbbbbbbbbbbbbbbbbb

`WRITE/OUT display.txt Para_1`
displays all the lines of the ASCII file display.txt which are found after a line beginning with Para_1 until a line beginning with ParaXYZ (XYZ may be any characters!) is found or until EOF, therefore it displays the line:
we all like weissbier

`WRITE/OUT display.txt Para xcf`
displays:
bbbbbbbbbbbbbbbbbbbbbbbbbbb

## WRITE/SETUP *core* 27-JAN-1994 KB

**Purpose:** Modify the variables of a Setup.

**Syntax:** `WRITE/SETUP [setup] [par1] [par2] [par3] ... [par_i]`

`setup` name of a Setup

`par1, par2, ...` parameters which depend upon the 'setup' chosen

**See also:** INITIALIZE/SETUP, READ/SETUP, INFO/SETUP

**Note:** Setups are a collection of variables (keywords) which have to be set up before certain, more complex commands can be executed.
You must use "INFO/SETUP abcdef" first to obtain the exact syntax of the WRITE/SETUP command for a specific Setup 'abcdef'.

**Examples:** `WRITE/SETUP catalog 2,4 2 NO 22,106`
Fill the variables of the Setup 'catalog' which are required for the command EXECUTE/CATALOG.
See "INFO/SETUP catalog" for an explanation of the syntax.

## WRITE/TABLE *core* 17-OCT-1983 JDP

**Purpose:** Store a value into one or several rows of a table

**Syntax:** `WRITE/TABLE table column row_sel value`

`table` the table name

`column` the column reference

`row_sel` the row(s) reference

`value` a constant. The value * will delete the element(s).

**See also:** COPY/KT

**Note:** The rows can be referenced to by using commas for enumeration and a double-dot for ranges, e.g. @1..5,10,15 designs the rows 1 to 10, the row 10 and the row 15. The command will expand the table, i.e increase its number of rows, if the row given in the command line exceeds the limits of that table. The command CREATE/ROW may be used to add rows at any position of the table. The descriptor HISTORY will not be updated by this command.

**Examples:** `WRITE/TABLE mytable :RADVEL @20..30 13.5`
insert the value 13.5 in the column :RADVEL of table mytable.tbl for the 11 rows 20, 21, .. 30 .

# XCORRELATE/IMAGE

## XCORRELATE/IMAGE *core* 08-MAY-1989 MR,KB

**Purpose:** Correlate 2 similar 1-dim frames over 2*(shift)+1 bandwith.

**Syntax:** `XCORRELATE/IMAGE temp spec result shift`

| | |
|---|---|
| `temp` | template frame |
| `spec` | frame to be analyzed |
| `result` | output frame to hold result |
| `shift` | 1/2 bandwith in integer pixels; defaulted to 10 |

**See also:** CONVOLVE/IMAGE, FFT/...

**Note:** If overlap too small (< 4 pixels) exit.
Identical NPIX,START,STEP descriptors (in the first dimension) of the input frames is required (and assumed).
This command is similar to the IHAP command SXCORRELATE.

**Examples:** `XCORRELATE/IMA a b res 15`
Correlate image 'b.bdf' with template 'a.bdf' over a total of 31 pixels, results are in image 'res.bdf'.

## ZOOM/CHANNEL *core* 30-APR-1990 KB

**Purpose:** Zoom image on image display.

**Syntax:** `ZOOM/CHANNEL [zoom_fact] [center]`

`zoom_fact` integer zoom factor;
if zoom_fact is omitted, zoom the image on image display according to size of rectangle formed by the two cursors (a);
maximum of the zoom factors in x and y is taken, so that equal magnification in x and y is assured;
if zoom factor is given, zoom the image with explicit zoom factor in interval [1,32] (b);
if zoom_factor = UP or DOWN (c), zoom up or down continuously:
every time the ENTER button is pressed the image is zoomed with the next higher (lower) zoom factor;
control returns to the terminal when highest (=32) or lowest (=1) zoom factor is reached, or if EXIT button is pressed;
defaulted to option (a)

`center` only for option (b) above:
x,y-screen coords. of center pixel for zooming;
if center is omitted, use cursor to choose center of zoom

**See also:** CLEAR/ZOOM, ZOOM/OVERLAY, GET/CURSOR, VIEW/IMAGE

**Note:** For all options (except (b) with explicit center values) you have to press the ENTER button to actually zoom. For option (a) the center of zoom is the center of the cursor rectangle otherwise the single cursor indicates the center.
Zooming is done in such a way that the zoom center pixel remains at the same spot (approximately).

**Examples:** `ZOOM/CHANNEL`
Modify size of cursor rectangle to determine the zoom factor and move it over the part of the image which should remain on display, press ENTER to magnify the image.

`ZOOM/CHANNEL 4`
Move cursor to pixel of which the position should remain fixed on display, press ENTER to magnify image by a factor of 4.

`ZOOM/CHANNEL 4 44,102`
Zoom image by a factor of 4, pixel at screen coordinates (44,102) remains (approx.) at same place on display.

# ZOOM/OVERLAY

## ZOOM/OVERLAY *core* 30-JUN-1991 KB

**Purpose:** Zoom image interactively and also the overlay plane.

**Syntax:** `ZOOM/OVERLAY [zoom_factor] [center]`

`zoom_fact` integer zoom factor;
if zoom_fact is omitted, zoom the image + overlay on image display according to size of rectangle formed by the two cursors (a);
maximum of the zoom factors in x and y is taken, so that equal magnification in x and y is assured;
this is the default;
if zoom factor is given, zoom the image + overlay with explicit zoom factor in interval [1,8] (b);
if zoom_factor = UP or DOWN (c), zoom up or down continuously:
Every time the ENTER button is pressed on the cursor box, the image + overlay is zoomed with the next higher (lower) zoom factor. Control returns to the terminal when highest (=8) or lowest (=1) zoom factors are reached, or if ENTER is pressed with cursor off.

`center` only for option (b) above: x,y-screen coords. of center pixel for zooming;
if center is omitted, use cursor to choose center of zoom

**See also:** CLEAR/ZOOM, ZOOM/CHANNEL, CLEAR/OVERLAY

**Note:** On workstations with X-Windows the overlay channel/plane is emulated in software. Therefore ZOOM/OVERLAY is not supported on workstations in MIDAS.

**Examples:** `ZOOM/OVER`
Modify size of cursor rectangle to determine the zoom factor and move it over part of image which should remain on display, press ENTER to magnify image + overlay.

`ZOOM/OVER 4`
Move cursor to pixel which should remain fixed on display, press ENTER to magnify image + overlay by a factor of 4.

# Appendix C

# Application Commands

# Application Commands

# ASSOCIATE/RANK

| ASSOCIATE/RANK | *applic* | 03-NOV-1989 | MP |
|---|---|---|---|

**Purpose:** Given two columns of a table, calculates the Rank-Order correlation coefficient (Spearman & Kendall)

**Syntax:** `ASSOCIATE/RANK table col1 col2 [action]`

| | |
|---|---|
| `table` | name of the table |
| `col1` | first column |
| `col2` | second column |
| `action` | K(endall) or S(pearman) defaulted to K |

**Note:** The command will return the Rank-Order correlation coefficient as well as the probability of exceeding its value under Null Hypothesis, the Null Hypothesis being:
"The two data sets are not associated "

**Examples:** `none`

BIN/TABLE *applic* 10-JAN-91 JEH

**Purpose:** Creates a table, bin.tbl with averages of col2 in bins of col1

**Syntax:** `BIN/TABLE table col1 col2 [bin] [min] [max] [sigma]`

| | |
|---|---|
| `table` | name of table file |
| `col1` | column to be binned (x-axis) |
| `col2` | column to be averaged (y-axis) |
| `bin` | if greater than zero: size of bin along col1 if less than zero : number of points in single bin |
| `min` | start-point of lowest bin, defaulted to minimum of col1 |
| `max` | end-point of highest bin, defaulted to maximum of col1 |
| `sigma` | points deviating more than sigma*standard_deviation will be excluded. |

**Note:** If [min] is larger than [max], the binning happens in reverse order (only relevant if [bin]<0).
There is a hard limit to the maximum number of entries in the table (16000), and to the maximum number of bins (1024).
The program creates an output table: bin.tbl, tables already existing with this name will be overwritten!!!! (UNIX). The format of this table is:
column 1: centre of bin
column 2: average of col2 in bin
column 3: standard deviation in the mean
column 4: width of distribution
column 5: number of points in bin

If [sigma] is specified, the bins are not rearranged, so bins may contain less points than specified by [bin] (negative number).
bugs: none sofar (?).

**Examples:** `BIN/TABLE mytable #2 #3 0.05 0.0 1.0`
Computes the averages of #3 in bins of 0.05 in #2, starting at 0.0 through 1.0

`BIN/TABLE mytable #2 #3 -5 1.0 0.0`
The same as above, only now each bin contains 5 points. Binning starts at 1.0, so the bin at 0.0 may contain less than 5 points.

`BIN/TABLE mytable #2 #3 -50 0.0 1.0 2`
Computes averages of #3 in bins containing 50 points. Then points deviating more than 2 sigma are excluded, and average and s.d. are recomputed. This way bins may contain less than 50 points

# COMPARE/2SAM

| COMPARE/2SAM | *applic* | 03-NOV-1989 | MP |
|---|---|---|---|

**Purpose:** Performs a Kolmogorov-Smirnov Two-sample Test on two columns of a table

**Syntax:** `COMPARE/2SAM table col1 col2`

| | |
|---|---|
| `table` | name of the table |
| `col1` | first column |
| `col2` | second column |

**Note:** The command will return the Kolmogorv-Smirnov D statistic as well as the probability of exceeding D under Null Hypothesis, the Null Hypothesis being : "The two data sets are drawn from the same distribution function" Warning : The probability becomes accurate when the size of the sample is large enough (>=20)

**Examples:** `none`

## COMPUTE/FIT *applic* 12-APR-1984 JDP

**Purpose:** Compute fitted values as an image of a column in a table. The user function is defined with the command EDIT/FIT, Function parameters are computed by the commands FIT/IMAGE or FIT/TABLE.

**Syntax:**

```
COMPUTE/FIT image[,error] [= function[(refima)]]
```

or

```
COMPUTE/FIT table y[,error] [= function[(ind)]]
```

In the first format :

| | |
|---|---|
| `image` | output image name |
| `error` | optional error mask |
| `function` | function name. Defaulted to the last used function. |
| `refima` | optional reference image defining the image space (start,step,no. of pixels). In the second format : |
| `table` | table name |
| `y` | output column, either already existing or a new one created by the command, to store the fitted values. |
| `error` | optional output column with the errors |
| `function` | function name. Defaulted to the last used function. |
| `ind` | optional column(s) to define the values of the independent variable(s) |

**Note:** In conextion to tables, the columns with the independent and dependent variables are in the same table.

**Examples:**

```
COMPUTE/FIT FITTED = FUNC
```

computes the image FITTED according to the parameter values in the function FUNC. Image space is defaulted to the domain used in the command FIT/IMAGE.

```
COMPUTE/FIT TABLE :FIT = FUNC(:X1,:X2)
```

computes the column :FIT in TABLE using the same parameter values as in the previous case, but the independent variables are defined in columns :X1 and :X2.

# COMPUTE/FUNCTION

## COMPUTE/FUNCTION *applic* 12-APR-1985 JDP

**Purpose:** Compute function values as an image of a column in a table. The function is a basic function or is defined as a linear combination of basic functions with the command EDIT/FIT.

**Syntax:** `COMPUTE/FUNCTION image = function[(refima)]`

or

`COMPUTE/FUNCTION table y = function[(ind)]`

| | |
|---|---|
| `image` | output image name |
| `function` | function name |
| `refima` | optional reference image defining the image space (start,step,no. of pixels). |
| | In the second format : |
| `table` | table name |
| `y` | output column, either already existing or a new one created by the command, to store the fitted values. |
| `function` | function name |
| `ind` | optional column(s) to define the values of the independent variable(s) |

**Note:** In connection to tables, the columns with the independent and dependent variables are in the same table. The command uses the guesses as actual parameters, while COMPUTE/FIT uses the fitted parameters. Basic functions can also be used, without need of previous editing (see last example). For a list of basic functions see chapter 5 of the manual.

**Examples:** `COMPUTE/FUNCTION IMAGE = FUNC`
computes IMAGE according to the parameter guesses in the function FUNC. Image space is defaulted to the domain used in the command FIT/IMAGE.

`COMPUTE/FUNCTION TABLE :VALUE = FUNC(:X1,:X2)`
computes the column :VALUE in TABLE using the same parameter values as in the previous case, but the independent variables are defined in columns :X1 and :X2.

`COMPUTE/FUNCTION IMAGE = GAUSS(REFERENCE;100.,10.,5.)`
computes IMAGE as a gaussian function, defined in the same domain as the image REFERENCE, with functional parameters : maximum value = 100., centre = 10., FWHM = 5.

## CREATE/FUNCTION *applic* 23-APR-1987 JDP

**Purpose:** Define user function(s) to be used by fitting commnands. The defined functions are coded in FORTRAN in files user00.for to user09.for. The command:
- compiles the functions;
- links the corresponding system primitives in the user work area;
- sets a system flag to execute the user version of the primitives.

The defined functions can be used in the EDIT/FIT command.

**Syntax:** `CREATE/FUNCTION fun1[,fun2...] [library_specs]`

`fun1[,fun2...]` function names as user00, ... user09

`library_specs` optional private libraries needed for linking

**Note:** User functions are coded as subroutines with the following arguments:
SUBROUTINE user0i(N, X, NP, P, Y, DY)
input arguments are : N integer*4 no. of independent variables X(N) real*4 array with the values of the independent vars. NP integer*4 no. of parameters P(NP) real*8 array with the values of the parameters output arguments are : Y real*8 output value of the function DY(NP) real*8 array with the values of the function derivatives with respect to each parameter.
IMPORTANT: the name of these files user0i.for HAS to be in lowercase.

**See also:** REPLACE/FUNCTION

**Examples:** `CREATE/FUNCTION user00,user03`
includes in the system the functions defined in the FORTRAN sources user00.for and user03.for

`CREATE/FUNCTION user00,user03 MYLIB/L,LIB3/L`
as above but link also with libraries mylib and lib3

# CREATE/GUI

| CREATE/GUI | *applic* | 05-DEC-1994 PB |
|---|---|---|

**Purpose:** Starts a graphical user interface for an application

**Syntax:** `CREATE/GUI [name]`

`name` Name of the required interface. Possible names are:

HELP : on-line documentation
DISPLAY : display related commands
LONG : context Long (1D and long-slit spectroscopy)
ALICE : spectral analysis context Alice

**Note:** a) The parameter name can be truncated to its shortest significant part
b) Color allocation problems could result from too many color-demanding aplications already created before the GUIs. In particular the display window or Unix applications like xv should preferably be created after the GUIs.
c) The command CREATE/GUI activates if necessary the contexts required for the interface.

d) Graphical user interfaces are available only for Unix systems and require OSF/MOTIF libraries at installation time.

**Examples:** `CREA/GUI disp`

CREATE/STAR *applic* 1-MRT-1994 RHW

**Purpose:** Create a frame containing the profile of the standard reference star, by adding and averaging inside a circular window some selected stars, after recentering

**Syntax:** `CREATE/STAR in_frame in_table out_frame [n_size] [frm_specs]`

`[dmin,dmax] [radius]`

| | |
|---|---|
| `in_frame` | input frame containing the selected stars |
| `in_table` | input table containing the coordinates of these stars in column :X_COORD and :Y_COORD |
| `out_frame` | output frame containing the calculated reference star |
| `n_size` | size of the output frame in number of pixels. The output frame will square, having n_size pixels in each dimension |
| `frm_specs` | specifications for the output frame in the format startx, starty, stepx, stepy. Default are the values of the input frame |
| `dmin,dmax` | low cut, high cut to select the stars; defaulted to the real cuts of the input frame |
| `radius` | in pixel number, the radius of a circular window centered on each selected star; default 20 |

**Note:** The input table containing the input star coordinates can easily obtained by the command GET/CURSOR.

The maximum of the stellar profile will be in the centre of the output frame.

**Examples:** `crea/star tspiral ttab ttest 128 0,0,0.1,0.1 ? 40`
Create a frame of 128 * 128 pixels containing the stellar profile obtained by averaging the stars stored in tabel 'ttab'. The input frame is 'tspiral'. The default for the low and high cuts is to be used; the radius of the circular window is 40 pixels.

# EDIT/FIT

## EDIT/FIT *applic* 12-APR-1984 JDP

**Purpose:** Interactive definition of functions. EDT compatible editing operation. The command creates and/or edits a structured array of functions and corresponding parameters, according to the rules defined in the notes below.

**Syntax:** `EDIT/FIT function`

`function` function to be created or modified. If the file function.fit does not exists, it is created by the command.

**Note:** The editor is a modified version of the EDT editor in keypad mode. The following keypad functions are not implemented : OPEN LINE, CHNGCASE, DEL EOL, CHAR, CUT, PASTE and SUBS. The following commands are implemented : EXIT and QUIT. On-line HELP facility available.
The editor uses a temporary table to store functions and parameters in two columns.
In the column FUNCTIONS the following info is stored: function specifications with parameter guesses . function specification : name(ind_var[,ind_var...];par1[,par2...])
In the column PARAMETERS the following info is stored: . parameter specification : par1=value – to define value and error par1=value[@] – to define value with the optional flag '@' to fix the parameter par1=constrain – to define parameter constrain as : constant*par2, par2*constant, par2/constant, or par2
A list of the functions supported by the system, and the description of the corresponding parameters is available in the printed version of the manual. Functions not supported by the system can be included with the command CREATE/FUNCTION.

**Examples:** `To fit two gaussians with equal fwhm and a linear background,`
the function edited with this command should look like :

```
POLY(X;A1,A2) A1=0.@ A2=2
```
– linear slope

```
GAUSS(X;B,C,D) B=10.  C=100.  D=4.
```
– first gaussian

```
GAUSS(X;E,F,G) E=15.  F=120.  G=D
```
– second gaussian

## FILTER/ADAPTIV *applic* 10-APR-1991, GMR

**Purpose:** Adaptive filtering of an image

**Subject:** Smoothing, gradient-filter, Laplace-filter

**Syntax:** `FILTER/ADAPTIV frame outframe [maskframe] [type] [shape] size k noise`

| | |
|---|---|
| `frame` | input image |
| `outframe` | result image |
| `maskframe` | input image for noise statistics: Only unmasked (mask-value = 0.) pixels of the input image are used to estimate the noise statistics. If 'NULL' is put in, no mask is used (default). |
| `type` | type of filter:<br>'S'= smoothing (default) 'G'= gradient-filter 'L'= Laplace-filter |
| `shape` | shape of impulse response:<br>'B'= box 'P'= pyramide (P is recommented and default) |
| `size` | maximal size of impulse response. In regions of high resolution the actual size is smaller. Possible sizes are: for box: 3,5,9,17,33,65,129 pixels,<br>for pyr.:3,5,7,11,15,23,31,47,63,95,127. |
| `k` | threshold for significance (see note). |
| `noise` | noise model:<br>'A'= additive noise assumed 'P'= Poisson-noise assumed |

**Note:** Algorithm: The local-signal-to noise ratio as a function of decreasing resolution is evaluated via the H-transform: mean gradients and curvatures over different scale lengths (obtained from the H-coefficients of different order) are compared to the corresponding exspectation values of the noise. The order for which this signal-to-noise ratio exceedes a given parameter k indicates the local resolution scale length of the signal (dubbed: the point becomes significant at this order), and determines the size of the impulse response of the filter at this point.
When the algorithm is finished some information on the noise statistics is printed out on the terminal: the standard deviation and the exspectation values of the gradients and the Laplace-termes at every order involved, and the the number of pixels which became significant on every order by the gradient and by the Laplace-term respectively. The rest pixels are set to the maximal size response.

**Examples:** `FILTER/ADAPTIV frame result null s p 31 3 a`
The image 'frame' is adaptively smoothed with a maximal filter sizs of 31x31 pixels. The noise statistics is estimated from the whole image (also signal is included! be careful when strong signal is in the image!). The image 'result' is displayed on channel 1.

# FIT/IMAGE

## FIT/IMAGE *applic* 20-FEB-1986 JDP,PhD

**Purpose:** Fit a user defined function to an image. The user function is defined with the command EDIT/FIT. See SET/FIT for modifying the method, weighting or printing qualifiers.

**Syntax:** `FIT/IMAGE [nfeval[,prec[,metpar]]] [image[,wgt]] [funct]`

| | |
|---|---|
| `nfeval` | maximum no. of function evaluations to be performed. If nfeval larger than initial values of the parameters are the user guesses nfeval smaller than initial values are taken from the previously computed parameter values |
| `prec` | precision to be achieved. |
| `metpar` | method parameters (different from one method to another, see manual for more details) |
| `image` | image to be fitted |
| `wgt` | optional weight or deviation of the measurements (see SET/FIT WEIGHT= ) |
| `funct` | is the user defined function name. By default the last used function is taken. |

**Note:** none

**Examples:** `FIT/IMAGE 10,0.0005,0.2 IMAGE TEST`
perform 10 iterations of the fit, with a relaxation factor 0.2 (for a METHOD=NEWTON), to reach a precision of 0.0005. Initial values of the parameters are the user guesses defined in the function TEST

`FIT/IMAGE -10,0.5,0.0005 IMAGE`
perform 10 iterations of the fit, using as initial parameters the values computed by the previous command.

## FIT/TABLE *applic* 06-JUNE-1986 JDP,PhD

**Purpose:** Fit an approximating function to a table column. The function is defined with the command EDIT/FIT. See SET/FIT for modifying the method, weighting or printing options.

**Syntax:** `FIT/TABLE [nfeval[,prec[,metpar]]] table dep,[wgt] ind [funct]`

| | |
|---|---|
| `nfeval` | maximum no. of function evaluations to be performed. If nfeval positive then initial values of the parameters are the user guesses nfeval negative then initial values are taken from the previously computed paramter values |
| `prec` | precision to be achieved. |
| `metpar` | method parameters (different from one method to another, see manual for more details) |
| `table` | table with the input data |
| `dep` | reference to the column with the dependent variable |
| `wgt` | optional reference to the column with the weight or deviation of the measurements (see SET/FIT WEIGHT=) |
| `ind` | reference to the column(s) with the independent variable(s) |
| `funct` | is the approximating function name. By default the last used function is taken. |

**Note:** none

**Examples:** `FIT/TABLE 10,0.0005,0.2 TABLE :Y :X1,:X2 TEST`
perform 10 function evaluation in the fit, with a relaxation factor 0.2 (if default have not been modified by SET/FIT i.e. METHOD = NEWTON), to reach a precision of 0.0005. Initial values of the parameters are the user guesses defined in the function TEST.

`FIT/TABLE -10,0.0005,0.5 TABLE :Y :X1,:X2`
perform 10 function evaluations in the fit, using as initial parameters the values computed by the previous command.

## FTEST/VAR *applic* 03-NOV-1989 MP

**Purpose:** Given two columns of a table, performs the F-Test for significantly different variances

**Syntax:** `FTEST/VAR table col1 col2`

| | |
|---|---|
| `table` | name of the table |
| `col1` | first column |
| `col2` | second column |

**Note:** The command will return the value of F as well as the probability of exceeding this value under the Null Hypothesis, the Null Hypothesis being : "The data sets have consistant variances"

**Examples:** `none`

# GET/FIT

## GET/FIT *applic* 28-AUG-1984 OGR

**Purpose:** Read the rectangular cursor to define subimages and initial guesses for fitting those with certain functions

**Syntax:** `GET/FIT table [image]`

| | |
|---|---|
| `table` | name of the table that is used to store the values provided by GET/CURSOR |
| `image` | optional image from which the pixel values are taken. By default the currently loaded image is taken. |

**Note:** The values :XSTART, :XEND, :YSTART and :YEND are used later on to extract as many subimages as entries are put into the table. Four values are then determined: :I0, :X0, :Y0 and :BACK. They are the maximum intensity in the subimage corrected for the background, the x and y where this maximum occurs and an estimate for the background intensity. The table will also contain the local averages in 3 x 3 pixel areas at the four corners of the subimages in columns :BACK_1 to :BACK_4.
On DeAnza Image Display the cursor box has to be set up as: both cursors on, TRACK off, RATE on.

**Examples:** `none`

## IDENTIFY/CURSOR *applic* 07-OCT-1985 JDP

**Purpose:** Identify table entries from image display system. This command : 1.finds entries in the table with the smallest distance (see note below) to the cursor position, and 2. asks for the identification (see note below)

**Syntax:** `IDENTIFY/CURSOR table ident x [y] [error]`

| | |
|---|---|
| `table` | table name |
| `ident` | reference to the column to be identified. If the column does not exist it is created with type 'R*4. |
| `x` | reference to the column with the abscisae |
| `y` | optional reference to the column with the ordinates |
| `error` | optional upper limit in the calculation of the distance |

**Note:** 1.The cursor box has to be set up as follows: Defined cursor on, TRACK off, RATE on. Pressing the ENTER button will list the table entry. Set defined cursor(s) off + press ENTER to exit. 2.The identification is entered from the keyboard as - a valid number of character string to be written into the table. - the symbol * to delete the identification. - RETURN to skip to the next cursor position. 3.The distance is the euclidean distance in one or two dimensions according to the reference to the ordinate column.

**Examples:** `IDENTIFY/CURSOR POSITIONS :IDENT :X :Y`

## IDENTIFY/GCURSOR *applic* 07-OCT-1985 JDP

**Purpose:** Identify table entries from the graphic display. This command : 1.finds entries in the table with the smallest distance (see note below) to the cursor position, and 2. asks for the identification (see note below)

**Syntax:** `IDENTIFY/GCURSOR table ident x [y] [error]`

| | |
|---|---|
| `table` | table name |
| `ident` | reference to the column to be identified. If the column does not exist it is created with type 'R*4. |
| `x` | reference to the column with the abscisae |
| `y` | optional reference to the column with the ordinates |
| `error` | optional upper limit in the calculation of the distance |

**Note:** For XWindows use the mouse to move the cursor, the leftmost mouse button or the RETURN key on the keyboard is the ENTER button: it will list the table entry. The second left mouse button serves as EXIT button.
The cursor cross may not be visible at first. Move the mouse to the lower left corner of the graphics window to grab the cursor.
For graphic terminals pressing any key of the graphics keyboard will list the table entry; the space bar is used to exit.
The identification is entered from the keyboard as:
- a valid number of character string to be written into the table. HENCE, it overwrites existing values !!!
- the symbol * to delete the identification.
- RETURN to skip to the next cursor position.

The distance is the euclidean distance in one or two dimensions according to the reference to the ordinate column.

**Examples:** `IDENTIFY/GCURSOR POSITIONS :IDENT :X :Y`

# INTEGRATE/APERTURE

## INTEGRATE/APERTURE *applic* 01-AUG-1990 RHW

**Purpose:** Compute integrated flux inside an aperture with a certain radius

**Syntax:** `INTEGRATE/APERTURE [in_specs] [out_tab] [radius]`

| | |
|---|---|
| `in_specs` | input specifications. Can be CURSOR, if the subimages are interactively chosen via the cursor rectangle, or frame, table if the subimages are defined in a table in the columns labelled :X_COORD, :Y_COORD. In the case no image display/window is needed. Default is CURSOR. |
| `out_tab` | output table if you want to store the results in a table The table will contain: x, y, radius, no of pixels, flux, background and magnitude in the columns :X_COORD, :Y_COORD, :RADIUS, :NPIX, :FLUX, :BGSB and :MAG. Default no output table is created. |
| `radius` | value in user unit to define the radius size of the aperture. This value can be defined for both input specifications. By default no radius is defined. This can only be used if the interactive mode is selected. |

**Note:** The command does not compute the background level. If descriptor 'BACKGROUND' exists the value of this descriptor is taken into account. In case the descriptor is non-existing the background is assumed to be 0.
The flux of the objects after sky background must be positive. If not the command will fail and will output zeros. I
The cursor operation depends on the image display in use. Consult Appendix D in the MIDAS Users Guide Volume A.

**Examples:** `INTEGRATE/APERTURE`
Use cursor to define subimages and display the results on the terminal only

`INTEGRATE/APERTURE CURSOR VALUES`
as above but write also the results in the table VALUES

`INTEGRATE/APERTURE CURSOR VALUES R`
as above but set the radius of the aperture to R

`INTEGRATE/APERTURE CCD001,SOURCES ? R`
use columns :X_COORD and :Y_COORD of table SOURCES (created by GET/CURSOR for example) to define the centers of the apertures compute the flux in the image CCD001 and just display results. NOTE : No display being used, the radius R is needed.

`INTEGRATE/APERTURE CCD001,SOURCES VALUES R`
as above but the results are stored also in the table VALUES

## INTEGRATE/LINE *applic* 30-APR-1990 RHW

**Purpose:** Integrate area a (spectral) line interactively via cursor input, or in batch, by interpolating polynomially between cursor positions.

**Syntax:** `INTEGRATE/LINE frame [y_coo] [x_sta,x_end] [n_cur,deg] [batch]`

`[x-pos,range]`

| | |
|---|---|
| `frame` | input frame name |
| `y_coo` | y-coordinate in frame (default @1). Both pixel number (@) or the world coordinate can be used. |
| `x_sta,x_end` | first and last point on the line to be displayed (either as @pixel_number or as real world coordinate). Defaults first and last image pixels. |
| `n_cur,deg` | no. of cursor positions (smaller than 100; default) and degree of the polynomial fit for the interpolation (default 1). |
| `batch` | B (for batch), anything else (for interactive use, default). In case the batch mode is used the third and fourth parameter are meaningless. |
| `x-pos,range` | independent world(!) coordinate positions along the x-axis, and the range around these positions within which the average pixel value is determined that will be used for the fit (default 0.0,0.0,0.0). |

**See also:** STATIST/IMAGE, READ/KEY

**Note:** The fluxes are computed by multiplication of the pixel intensities and the pixel separation. If a cursor position falls between two frame pixels a correction is applied to include the flux in this pixel fraction in the total flux. The correction is equal to the size of the pixel fraction times the pixel intensity.
For each integration the results, x-start and x-end (pixel and world coordinates), step size (pixel separation), total flux, cont. flux, line flux, fraction line to continuum, and equivalent width are written into the first 10 locations of the keyword OUTPUTR.
Interactive use: Use any key except (incl. RETURN), or the mouse enter button to validate the cursor (both x and y) positions. When the specified number of cursor positions has been entered, the interpolated line is drawn and the integration is done. Use the space bar or the mouse exit button to terminate.
Batch use: No graphics terminal required; just give a 'B' value for the batch parameter, and relevant x-values and range. Contrary to the interactive integration, where the y-positions of the cursor are used to determine in the intergration interval, in batch mode the y-positions are equal to the pixel values of the corresponding input x-coordinates. If the x-coordinate falls between two frame pixels the y-position is obtained from a straight average of both pixel values.
In case the interactive mode is used, the maximum number of pixels that can be plotted is 100000, in case bin mode is on, 50000. A fatal error occurs if this number is exceeded.
Note, that for batch use, the third and fourth parameter (x-start, x-end and n_cur,deg) are irrelevant.

**Examples:** `INTEGRATE/LINE GL @1 5,50 2,1`
Do the integration(s) on line 1 of frame GL between world coordinates 5 and 50, using two cursor positions and linear interpolation.

`INTEGRATE/LINE GL @1 ? ? B 23.5,26.5,0.05`
Batch version, with x-coordinates (and tolerance) specified. Look at the output: READ/KEY OUTPUTR.

# INTEGRATE/STAR

## INTEGRATE/STAR *applic* 31-JUL-1990 RHW

**Purpose:** Computes flux, radius and background of stars previously centered

**Syntax:** `INTEGRATE/STAR [in_specs] [out_table] [parameters] [mode]`

| | |
|---|---|
| `in_specs` | input specifications. Can be CURSOR, if the objects are interactively chosen with the display cursor, or frame,table if the objects are defined in a table. In the latter case no image display/window is required. The table should have two columns labeled X_COORD and Y_COORD. It can be created by the commands GET/GCURSOR and GET/CURSOR. Default input is CURSOR. |
| `out_table` | output table if you want to store the results in a table. The table will contain: x, y, flux, magnitude, and background in the columns :X_COORD, :Y_COORD, :FLUX, :BGSB, and :MAG. Default no output table will be created, and results appear on the screen. |
| `parameters` | radius,step,ref_magnitude (defaulted to 20,1,0), where radius is the maximum radius in pixels of the aperture; step is the step in pixels between 2 annuli; ref_magnitude is the magnitude of the reference star. |
| `mode` | AUTO or INTERACTIVE; only applicable if in_spec is given frame,table as input. default INTERACTIVE. In the interactive mode the graphics display/window should be available. |

**Note:** The command computes the (stellar) object magnitude according to the following receipe. The algorithm is based on determination of the parameters in a single line through the (stellar) object, not on a two-dimenional approch.

1. First, the image pixel values are sorted and put in a table, running from low to high values.
2. From the lowest 4 values in this table a first order average sky background and standard deviation (sigma) is computed;
3. Then, the algorithm looks for the table value which falls between sky average and average + sigma/2;
4. Depending where in the table this value is found the average sky value and sigma are adjusted to determine the final values;
5. The radius of the object is then computed considering only those points above sky average + 6*sigma.

Because the one-dimensional approch results obtained from less than 10 pixel values should be considered to be unreliable. In most case a error message will be given.

**Examples:** `INTEGR/STAR ? rfotout`

Compute interactively the fluxes in frame loaded using the image and graphics display. The command will run in INTERACTIVE mode.

`INTEGR/STAR rfot,rfotpos rfotout 5,1,0 AUTO`

Use columns :X_COORD and :Y_XCOORD of the table rfotpos (e.g. created by GET/CURSOR) to define the centers of the objects for which the fluxes are to be computed. The results will be stored in the table rfotout. The command runs in automatic mode and uses non-default values for the radius, step, and reference magnitude.

## KSTEST/1SAM *applic* 03-NOV-1989 MP

**Purpose:** Performs a Kolmogorov-Smirnov One-sample Test .

**Syntax:** `KSTEST/1SAM table col [distri] [coeffs]`

| | |
|---|---|
| `table` | name of the table |
| `col` | column containing the set of sample values. |
| `distri` | theoritical distribution<br>U(NIFORM) for uniform probability function<br>G(AUSS) for Gaussian p.f.<br>E(XPONENTIAL) for exponential p.f.<br>P(OISSON) for Poisson p.f.<br>distri is defaulted to UNIFORM |
| `coefs` | coefficients for the distribution above, number and meaning depends on the parameter distri, defaulted to 0.,1. |

**Note:** The command provides a test of the Null Hypothesis:
"The data set comes from a distribution having the theoretical distribution" against the hypothesis:
"The data set cannot be considered to be a random sample from the specified theoritical distribution" The command will return the Kolmogorov-Smirnov D statistic (largest absolute deviation between the sample cumulative distribution and the theoritical distribution) as well as the probability of exceeding D under Null Hypothesis,

**Examples:** `none`

## MODIFY/FIT *applic* 29-AUG-1984 JDP

**Purpose:** Modify parameter values in a fit file. The value corresponding to parameter 'par' is taken from column ':par_GUESS' in the given table. The command can be used in relation with SAVE/FIT

**Syntax:** `MODIFY/FIT table seq [name]`

| | |
|---|---|
| `table` | name of table file |
| `seq` | seq. number as '@n' |
| `name` | fit name |

**Note:** none

**Examples:** `The following set of commands are used to fit`
a function with different initial guesses:

```
MODIFY/FIT MYTABLE @1 FIT
```
modify parameters

```
FIT/IMA -10,0.1 IMAGE
```
use last initialized parameters

```
SAVE/FIT MYTABLE @1 FIT
```
save adjusted parameters

# PRINT/FIT

## PRINT/FIT *applic* 12-APR-1984 JDP

**Purpose:** Print fitted values of the function parameters

**Syntax:** `PRINT/FIT func_name`

`func_name` name of the fitted function

**Note:** none

**Examples:** `PRINT/FIT FUNCTION`

## READ/FIT *applic* 12-APR-1984 JDP

**Purpose:** Display fitted values of the parameters of the given function

**Syntax:** `READ/FIT func_name`

`func_name` name of the function used in the fitting process

**Note:** none

**Examples:** `READ/FIT FUNCTION`

REGISTER/SESSION *applic* 19-JAN-1994 PB

**Purpose:** Called at the initialisation of a context, the command REGISTER/SESSION creates a keyword table and sets internal keywords to register a session. This operation allows to define session management commands (e.g. SET/.. KEYDEL/.. HELP/.. SAVINIT/.. etc..) which behavior is identical in all packages using the session manager.

**Syntax:** `REGISTER/SESSION session directory file table`

| | |
|---|---|
| `session` | identification name of the session<br>(e.g. ccd, long, echel, ...). The two first letters are used for identification of the session. |
| `directory` | MIDAS name of the directory where the initialisation file is located.<br>(e.g. STD_PROC, CON_PROC). |
| `file` | Name of the initialisation file, complete with extension .prg. |
| `table` | Name of the output keyword table, complete with extension .tbl. |

**See also:** INQUIRE/SESSION, WRITE/SESSION

# REGISTER/SESSION

**Note:** 1) The command REGISTER/SESSION parses the initialisation file to generate the output keyword table. The first and last parsed lines can be indicated by the two comments lines:
!Begin Session List
!End Session List
which must be written exactly as above, the sign ! corresponding to the first character of the line.

2) Within the parsed section of the initialisation file, each keyword declaration line must have the following structure:
WRITE/KEY <key_def> <default> ! <Description>
(e.g. WRITE/KEY CCDNAME/C/1/60 "?" ! CCD Name)

All items must be present. Lines which first non blank character is an exclamation mark (!) will be ignored.

3) When a session is registered, a list of session management commands can be defined within the context, including:

HELP/<session> key [mode]
provides information about session keywords. Information can be looked by selection of the keyword list (mode=KEY, default mode), or by selection of the description list (mode=DES).

SET/<session> key1 = value1 key2 = value2 ...
assigns a value to the session keywords. Keyword names can be truncated to the shortest non ambiguous part. Blank characters before and after the = sign are optional. Multiple keywords can be set with the syntax, e.g.:
SET/LONG NPIX=,340
corresponding to NPIX(2) = 340.

SAVINIT/<session> file [mode]
allows all keywords of a session to the saved as or retrieved from the descriptors of a MIDAS file (image or table). Mode can be WRITE (default) or READ.

KEYDEL/<session>
allows all keywords of a session to be deleted.

ERROR/<session> command key
displays a standard message of the format:
<command>: Wrong parameter <key> = <key>
and displays the help related to this sesion keyword.

4) The above commands must be declared in the context file as follows:

CREATE/COMM SET/<session> @a keyset CREATE/COMM SAVINIT/<session> @a keyrdwr CREATE/COMM HELP/<session> @a keyhelp CREATE/COMM KEYDEL/<session> @a keydel CREATE/COMM ERROR/<session> @a keyerror
where <session> is the name of the declared session, like CCD, LONG, ECHELLE, MOS, etc...

**Examples:** `REGISTER/SESSION long STD_PROC lninit.prg lntab.tbl`

## REPLACE/FUNCTION *applic* 07-JULY-1986 PhD, JDP

**Purpose:** Replace user function(s) to be used by fitting package. This command has to be used after CREATE/FUNCTION when a modification is needed in the user defined function. The defined functions are coded in FORTRAN in files user00.for to user09.for. The command
- compiles the functions,
- links the corresponding system primitives in the user work area,
- sets a system flag to execute the user version of the primitives.

The defined functions can be used in the EDIT/FIT command.

**Syntax:** `REPLACE/FUNCTION fun1[,fun2...]`

`fun1[,fun2...]` function names as user00, ... user09

**Note:** User functions are coded as subroutines with the following arguments:

SUBROUTINE user0i(N, X, NP, P, Y, DY)
input arguments are :
N integer no. of independent variables
X(N) real array with the values of the independent vars.
NP integer no. of parameters
P(NP) double precision array with the values of the parameters
output arguments are :
Y double precision output value of the function
DY(NP) double precision array with the values of the function derivatives with respect to each parameter.
IMPORTANT: The name of these files user0i.for HAS to be in lowercase.

**Examples:** `CREATE/FUNCTION user00,user03`
includes in the system the functions defined in the FORTRAN sources user00.for and user03.for after modifications in user03.for, the command REPLACE/FUNCTION user03 replaces in the system this function.

# SAVE/FIT

## SAVE/FIT *applic* 29-AUG-1984 JDP

**Purpose:** Save fitted parameter values and errors in a table. The value corresponding to parameter 'par' are stored in column ':par' and the error in ':par_ERROR' The command can be used in relation with MODIFY/FIT

**Syntax:** `SAVE/FIT table seq [name]`

| | |
|---|---|
| `table` | name of table file |
| `seq` | seq. number as '@n' |
| `name` | fit name |

**Note:** none

**Examples:** `The following set of commands are used to fit a function with` different initial guesses:

`MODIFY/FIT MYTABLE @1 FIT`
modify parameters

`FIT/IMA -10,0.1 IMAGE`
use last initialized parameters

`SAVE/FIT MYTABLE @1 FIT`
save adjusted parameters

## SELECT/FUNCTION *applic* 12-JUNE-1985 JDP

**Purpose:** select function components

**Syntax:** `SELECT/FUNCTION name number[,...]`

| | |
|---|---|
| `name` | function name, as given in the EDIT/FIT command |
| `number[,...]` | component number(s), defined by number or by the key ALL |

**Note:** The selected function components will be used by the commands COMPUTE/FUNCTION and COMPUTE/FIT

**Examples:** `SELECT/FUNCTION MYFUNC 1,3`

select components 1 and 3

`COMPUTE/FIT OUT = MYFUNC`

compute fitted values using these two components

`SELECT/FUNCTION MYFUNC ALL select all components`

## SET/FIT *applic* 02-SEP-1992 PD+JDP+MP

**Purpose:** Modify qualifiers of the FIT command

**Syntax:** `SET/FIT par=value [par=value ...]`

`par` parameter name

`value` parameter value
Allowed parameters and values are:
METHOD - fitting method as
NR : Newton-Raphson - derivatives required (default)
CGNND : Corrected Gauss-Newton method - no derivatives
QN : Quasi Newton - derivatives required
MGN : Modified Gauss-Newton - derivatives required
PRINT - integer number to control the printing of the intermediate fitted values. If negative, the correlations matrix will be computed and displayed. Default is 1.
WEIGHT - weighting scheme as
C : Constant (=no) weighting (default)
W : Weight is given in the FIT command (consult also HELP on FIT)
I : Instrumental weighting w(i)=1/sigma(i)**2 (consult also HELP on FIT)
S : Statistical weighting w(i)=1/indep_var(i)

FUNCT - FIT function to be used. any function created by EDIT/FIT. Default name is FIT.
FCTDEF - function definition type SY[ST] : Only MIDAS defined functions are used (Default). US[ER] : User functions (USER_) are defined.
BOUNDS - parameters bounds type N : No bounds (default) P : Parameters are forced to be positive I : Bounds are provided by the user in a table . The name of the table can be defined by the parameter BNDTAB. The table should contain two columns, the first one for the lower bound, the second one for the upper bound and as many rows as the number of parameters. G : The same bounds are used for all parameters. There will be read from the first row of the table defined by the parameter BNDTAB.

BNDTAB - parameter bounds table

**Note:** none

**Examples:** `SET/FIT METHOD=QN PRINT=2`
or SET/FIT QN 2 select QN method and print intermediate results every second iteration. No correlations computed.

## SHOW/FIT *applic* 20-FEB-1986 PD + JDP

**Purpose:** Show parameters and current function used in the FITTING package.

**Syntax:** `SHOW/FIT`

**Note:** none

**Examples:** none

# SORT/COLUMN

## SORT/COLUMN *applic* 19-JUN-1991 PB

**Purpose:** Sort pixel values of a frame along columns

**Syntax:** `SORT/COL input output`

`input` name of input frame

`output` name of output frame

**Note:** This command enables to correct CCD images for bad columns patterns by the following procedure:

If ima_org is the science frame containing bad columns, and ncol, nrow are respectively the number of columns and of rows in ima_org. It is possible to prepare a mask of the bad columns using:

SORT/COL ima_org ima_sort

AVERAGE/ROW mask1d = ima_sort @<row1>,@<row2>

where row1, row2 delimit an area of the sorted image containing only pixels due to the background of ima_org.

GROW/IMAGE mask2d = mask1d 1.,1.,<nrow>

where nrow is the number of rows in ima_org. Descriptors start and step of ima_org are suppposed to be equal to 1.,1.

COMPUTE/IMA ima_cor = ima_org - mask2d

This method is based on Pojmanski (1991, How To Remove Easily the Unpleasant Column Pattern, Proceedings of the 3rd ESO/ST-ECF Data Analysis Workshop, pp. 101-105)

**Examples:** `none`

## SORT/ROW *applic* 19-JUN-1991 PB

**Purpose:** Sort pixel values of a frame along rows

**Syntax:** `SORT/ROW input output`

| | |
|---|---|
| `input` | name of input frame |
| `output` | name of output frame |

**Note:** This command enables to correct CCD images for bad rows patterns by the following procedure:

If ima_org is the science frame containing bad rows, and ncol, nrow are respectively the number of columns and of rows in ima_org. It is possible to prepare a mask of the bad rows using:

SORT/ROW ima_org ima_sort

AVERAGE/COL mask1d = ima_sort @<col1>,@<col2>

where col1, col2 delimit an area of the sorted image containing only pixels due to the background of ima_org.

GROW/IMAGE mask2dr = mask1d 1.,1.,<ncol>

where ncol is the number of columns in ima_org. Descriptors start and step of ima_org are suppposed to be equal to 1.,1.

ROTATE/COUNTER mask2dr mask2d WRITE/DESCR mask2d start 1.,1. WRITE/DESCR mask2d step 1.,1.

COMPUTE/IMA ima_cor = ima_org - mask2d

This method is based on Pojmanski (1991, How To Remove Easily the Unpleasant Column Pattern, Proceedings of the 3rd ESO/ST-ECF Data Analysis Workshop, pp. 101-105)

**Examples:** `none`

## STEST/MEAN *applic* 03-NOV-1989 MP

**Purpose:** Given two columns of a table, performs the Student t-test for significantly different means

**Syntax:** `STEST/MEAN table col1 col2`

| | |
|---|---|
| `table` | name of the table |
| `col1` | first column |
| `col2` | second column |

**Note:** The data sets are assumed to be drawn from populations with the same variance. The command will return the value of Student's t as well as the probability of exceeding its value under the Null Hypothesis, the Null Hypothesis being :
"The data sets have consistant means"

**Examples:** `none`

# TUTORIAL/ALIGN

## TUTORIAL/ALIGN *applic* 21-DEC-1994 KB

**Purpose:** Explain the alignment of two images.

**Syntax:** `TUTORIAL/ALIGN inputs result align_option`

| | |
|---|---|
| `inputs` | inima,refima name of image to be aligned and name of reference image;<br>or AUTO for automatic mode, in that case images are copied from the MID_TEST directory to serve as input for the tutorial;<br>if omitted, the user is asked to enter the names interactively |
| `result` | name of result, i.e. aligned, image;<br>if omitted, the user is asked to enter the names interactively |
| `align_option` | the 'option' parameter (par. 3) of the ALIGN/IMAGE command; defaulted to UNIT (as in ALIGN/IMAGE) |

**See also:** ALIGN/IMAGE, REBIN/ROTATE, CENTER/GAUSS

**Note:** In the course of the tutorial the input image is loaded into a Midas display window and you have to click on different stars in the display to obtain rectangles around them which are used to calculate their centers. If the reference image is only slightly misaligned, i.e. the same stars in the ref-image fall also inside these rectangles, this table can also be used to compute the corresponding centers in the ref-image. Otherwise, also the ref-image is loaded into the display and the rectangles around the corresponding stars have to be entered again in the SAME order!
These tables are then used in an ALIGN/IMAGE command and the results of that provide the input for the final REBIN/ROTATE command.

**Examples:** `TUTORIAL/ALIGN AUTO`
Run the tutorial in automatic mode.

`TUTORIAL/ALIGN ? ? free`
Enter the names of input and result frames interactively, use the ALIGN/IMAGE command with option=FREE .

## TUTORI/FIT *applic* 24-OCT-1994 MP

**Purpose:** explains the modelling of table and image data by fitting non-linear functions.

**Syntax:** `TUTORIAL/FIT`

See also: COMPUTE/FIT, COMPUTE/FUNCTION, EDIT/FIT, FIT/IMAG, FIT/TABL, MODIFY/FIT, REPLACE/FUNCTION, SAVE/FIT, SELECT/FUNCTION, SET/FIT, SHOW/FIT

**Note:** None

**Examples:** `TUTORIAL/FIT`
Run the tutorial.

# Appendix C

# Application Commands

# Appendix D

# Standard Reduction - echelle

# Standard Reduction Commands

# Context: echelle

# AVERAGE/TABLE

## AVERAGE/TABLE *stdred/echelle* 17-AUG-1992 PB

**Purpose:** The pixel values of the frame are read at positions defined by the columns xy_col in the table. The values are written in the column outcol.

**Syntax:** `AVERAGE/TABLE frame table xy_col outcol [size]`

| | |
|---|---|
| `frame` | input image. |
| `table` | table name. |
| `xy_col` | name of the columns defining the world coordinates X and Y of the pixels to read. |
| `outcol` | name of the column in which the pixel values are written. |
| `size` | Half-size of the squared area in which the flux is measured. Default is 0. |

**Note:** Any selection set in the input table will be respected and the relevant positions ignored.

**Examples:**
```
MERGE/ECHELLE extr1 spec1 3.  AVERAGE
```

## BACKGROUND/ECHELLE *stdred/echelle* 29-JUL-1991 PB

**Purpose:** Compute the background of the echelle raw spectrum in the sampling space pixel-pixel. This command requires the table Background values are estimated at reference positions between the orders and interpolated either by smoothing spline or bivariate polynomial interpolation. Commands SCAN/ECHELLE and SELECT/BACKGROUND can be used before.

**Syntax:** `BACKGROUND/ECHELLE in out [radx,rady,[step]] [deg] [smooth] SPLINE`

`BACKGROUND/ECHELLE in out [degx,degy,[step]] P6=POLY (see note)`

| | |
|---|---|
| `input` | input raw spectrum, sampled in pixel-pixel space |
| `output` | computed background image |
| `radx,rady` | radius of the window for median estimate of the background. The quantity 2*rady+1 must be smaller than the interorder width. The window is tilted to follow the orders, so that the distance 2*radx+1 is parallel to the orders.<br>Echelle keyword is BKGRAD; it is shown in SHOW/ECHEL if BKGMTD=SPLINE. Default value is 6,2. |
| `degx,degy` | Degrees of the bivariate polynomial in method POLY. Echelle keyword is BKGPOL and is displayed by SHOW/ECHELLE if BKGMTD=POLY. Default value is 2,3. |
| `step` | step along X axis between background reference positions. Echelle keyword is BKGSTEP. |
| `deg` | Degree of the spline polynomials used to define the background values in method SPLINE.<br>degree=0 is not allowed<br>Echelle keyword is BKGDEG; is displayed by SHOW/ECHE if BKGMTD=SPLINE. Default is 3. |
| `smooth` | Value of the smoothing factor in SPLINE method. Echelle keyword is BKGSMO and is displayed by SHOW/ECHE if BKGMTD=SPLINE. Default is 1000. |
| `method` | POLY or SPLINE. Echelle keyword is BKGMTD, default is POLY. |

**See also:** BACKGR/SMOOTH, SCAN/ECHELLE, SELECT/BACKGROUND, SUBTRACT/BACKGR

# BACKGROUND/ECHELLE

**Note:** This command enables to choose between two methods to estimate the interorder background. The selection of the method depends on the value of parameter P6 or echelle keyword BKGMTD (see syntax).

1) Bivariate Polynomial Method

A bivariate polynomial is fitted to the values of interorder background measured at positions defined by columns :X and :YBKG of the table back.tbl.

Parameters degX and degY are respectively the degree of the polynomial along X and Y axis. Defaults are degX=2 and degY=3.

Null degrees are allowed by polynomial method

2) Spline Method

The background is approximated using a smoothing spline algorithm. Reference positions of the background are defined in columns :X and :YBKG of the table back.tbl created by DEFINE/ECHELLE or DEFINE/HOUGH.
Only the selected positions of the table back.tbl are used to estimate the background. It is possible to unselect positions with the command SELECT/BACK.

The background level on each of these reference positions is defined as the median value estimated on a window which radius is radx,rady. The estimated values are written in column :BKG of table middummb.tbl.

The values are then approximated using a smoothing spline algorithm. The smoothing factor allows to interpolate the values (smooth=0.) or to smooth them (the optimal smoothing factor depends on the standard-deviation of the background values).

For more details on how to optimize background parameters, see MIDAS documentation, Vol. B, Chapt. 8: Echelle Package.

Bugs
The descriptor HISTORY of the output frame is the same as the one of the input frame, except for the last line, which is overwritten by a message describing the background computation parameters.

**Examples:**

```
BACKGROUND/ECHELLE obj backobj 10,2,5 3 0.  SPLINE

BACKGROUND/ECHELLE obj backobj P6=SPLINE

BACKGROUND/ECHELLE obj bak 3,4 P6=POLY
```

BACKGROUND/SMOOTH *stdred/echelle* 25-JUL-1991 PB

**Purpose:** Compute the background of the echelle raw spectrum in the sampling space pixel-pixel using filtering and pixel replacement. This method does not require the table order.

**Syntax:** `BACKGROUND/SMOOTH input output [radx,rady] [niter] [visu]`

| | |
|---|---|
| `input` | input raw spectrum, sampled in pixel-pixel space |
| `output` | computed background image |
| `radx,rady` | radius in x and y for smoothing. rady must be somewhat larger than the order width. Echelle keyword is BKGRAD, displayed by SHOW/ECH if BKGMTD=SMOOTH. Default: 5,30 |
| `niter` | number of iterations. Echelle keyword is BKGNIT, displayed by SHOW/ECH if BKGMTD=SMOOTH. Default: 10. |
| `visu` | Possible values are YES or NO. If visu=YES a graphic and a display window are created if necessary. The graphic window shows the background estimated for the central trace of the input image. The display window shows output image. Cuts value are read from LHCUT descriptor of input.<br>Echelle keyword is BKGVISU, displayed by SHOW/ECHE if BKGMTD=SMOOTH. Default: YES. |

**See also:** BACKGROUND/ECHELLE, SCAN/ECHELLE, SUBTRACT/ECHELLE

**Note:** It is advised to properly set the keyword SCAN before running this command by using the command SCAN/ECHELLE.

At the difference of the related command BACKGR/ECHELLE this command does not require the table order created by DEFINE/ECHELLE or DEFINE/HOUGH.

**Examples:**
```
BACKGROUND/SMOOTH ccd0008 bkg0008
```

# CALIBRATE/ECHELLE

## CALIBRATE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Perform the initial steps of an echelle reduction, i.e. a) trace orders and create tables order.tbl and back.tbl b) extract wavelength comparison spectrum and create table line.tbl c) perform the wavelength calibration.
This command combines several lower-level commands. It is strongly recommended that echelle beginners familiarize themselves with them:
DEFINE/ECHELLE, DEFINE/HOUGH
EXTRACT/ECHELLE
SEARCH/ECHELLE
IDENT/ECHELLE
PLOT/RESIDUAL
INI/ECH, SET/ECH, SHO/ECH, HELP/ECH (see note below)

**Syntax:** `CALIBR/ECH [defmtd] [wlcmtd]`

| | |
|---|---|
| `defmtd` | order definition method as<br>STANDARD : if order width is smaller than the interorder space<br>COMPLETEMENT : if order width is greater than the interorder space<br>HOUGH : to perform the order detection by Hough transform. |
| `wlcmtd` | wavelength calibration method as:<br>PAIR : interactive identification of two lines in overlapped regions of the spectrum.<br>ANGLE : interactive identification of four lines.<br>GUESS : to use a previous session as a guess. (specified by SET/ECH GUESS=...)<br>RESTART : to reuse last pointed lines<br>ORDER : to recompute single-order solutions with a new TOL parameter. |

**Note:** Most of the parameters required to control this command are set and checked using the SET/ECHE, SHOW/ECHE, HELP/ECHE commands. The sections of the SHOW/ECHELLE that must be checked are:
1. Order definition
4. Extraction
5. Lines searching
6. Wavelength calibration

**Examples:**
```
CALIBRATE/ECHELLE

SET/ECHELLE WLCMTD=ANGLE WLCLOOP=0.5,1.5,10

CALIBRATE/ECHELLE

SET/ECHELLE WLCMTD=GUESS GUESS=casp

CALIBRATE/ECHELLE
```

## CLEAN/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** Clears contexts Echelle and Spec and removes process tables. Session keywords are also deleted.

**Syntax:** `CLEAN/ECHELLE`

**See also:** CLEAR/CONTEXT, SET/CONTEXT, INIT/ECHELLE, INIT/EMMI

**Note:** None

**Examples:** `CLEAN/ECHELLE`

## CONVERT/ECHELLE *stdred/echelle* 17-DEC-1991 PB

**Purpose:** Resample echelle orders. The command is used internally by RESPONSE/ECHELLE and REPEAT/ECHELLE to convert the instrument response from high to low resolution and vice versa.

**Syntax:** `CONVERT/ECHELLE input output domain function param option`

| | |
|---|---|
| `input` | input file with the echelle orders (wavelength-order format) |
| `output` | resampled output file |
| `reference` | defines the output sampling domain as - two real numbers: start,step - a filename used as reference |
| `function` | rebinning function (LIN,POL,INV,EXP,DEX,LOG,DLG,IPO,U01) |
| `param` | parameter related to the conversion method (for LIN, default is 0.,1.) |
| `option` | rebinning method. (PIX,LIN,SPG) |

**Examples:** `CONVERT/ECHELLE INPUT OUTPUT REFERENCE LIN 0.,1. SPG`

# DEFINE/ECHE

## DEFINE/ECHE *stdred/echelle* 29-JUL-1991 PB

**Purpose:** Search for order positions in a 2-D echelle frame. The positions of points located in order and in interorder space are stored in an auxiliary table, named order.

**Syntax:** `DEFINE/ECHE [ordref] [width1,thres1,slope] [defmtd] [defpol]`

| | |
|---|---|
| `ordref` | raw flat field used as input image Echelle keyword is ORDREF. |
| `width1` | is the full width at half maximum of the orders accross the dispersion direction (in pixels). Echelle keyword is WIDTH1 and default is 8. |
| `thres1` | is the absolute threshold for the order detection (see note). Echelle keyword is THRES1 and default value is 1000. |
| `slope` | initial estimate of the mean slope of the orders. If the spectra have been correctly rotated to the standard orientation (see ROTATE/ECHELLE), the slope must be slightly positive. Echelle keyword is SLOPE and default value is 0.05 |
| `defmtd` | defines the method used to detect the orders. Possible methods are :<br>STD: the standard (default) method, recommended for well separated orders (e.g., for CASPEC above 4700 A).<br>COM: thresholding is done after complementing pixel values. Useful where orders are more crowded (interorder space less than order width).<br>Echelle keyword is DEFMTD. Default is STD. |
| `defpol` | degree of the 2D polynomial used to approximate the position of the orders as y = f(x,m), where<br>y: is the line number in pixels;<br>x: is the sample number in pixels;<br>m: a relative order number.<br>Echelle keyword is DEFPOL. Default is 3,4 |

**See also:** DEFINE/HOUGH.

**Note:** The auxiliary table order is used as intermediate storage Parameters can be defaulted to the corresponding session values (SHOW/ECHELLE to display the current values).

The performance of the algorithm depends sensitively on the value entered for the width of the order profile.

In its present version, the algorithm will crash if the exposure level of a detected order drops below the detection threshold. This can cause problems if, as can be the case with the Echellec, in the center of the frame the scattered light level in the interorder region is higher than the exposure level within the orders towards their end(s). Use `EXTRACT/IMAGE` to extract the useful part of the frame.

**Examples:** `DEFINE/ECHELLE stdframe 6,10`

DEFINE/HOUGH *stdred/echelle* 7-DEC-1994 PB

**Purpose:** Search for order positions in a 2-D echelle frame. The positions of points located in order and in interorder space are stored in an auxiliary table, named order.tbl

**Syntax:** `DEFINE/HOUGH [ordref] [nbord] [hwid] [hough_par] [thresh]`

`[degx,degy] [hot_thres,step] [hough_setup]`

`ordref` raw flat field used as input image.
Echelle keyword is ORDREF.

`nbord` number of order to be detected.
If nbord is set to 0, the number of orders is estimated automatically.
If nbord<0, the procedure will detect abs(nbord) orders, then prompt for a differential number of orders (e.g. +3 for 3 more orders, -1 for one less), until the user enters 0 to finish the procedure.
Echelle keyword is NBORDI. Default value is 0.

`hwid` half-width of the orders in pixel. If hwid is set to 0, the half-width is measured for each order.
Echelle keyword is WIDTHI. Default value is 0.

`hough_par` Possible values are DENSE, CENTER, ALL, NO, FOLLOW, or Step,Nb-trace. Default is DENSE. No echelle keyword controls this parameter. If it is preceded by a minus sign (e.g. -CENTER), no median filter is applied to the frame.

DENSE generate a Hough transform based on 50 columns evenly distributed on the frame.

CENTER generate a Hough transform based on 50 columns covering 25% of the frame width in its central part.

ALL generate a full Hough transform of the frame.

step,nb-trace generate a Hough transform based on (nb-trace) columns separated by (step) pixels around the central column.

NO skips the Hough transform and take frame middummh.bdf as the hough transform. &h is created by a previous DEFINE/ECHELLE with option hough-par = DENSE,ALL or Step,nb-tr.

FOLLOW skips the Hough transform and order detection and performs only order following and bivariate fitting.

`thresh` threshold for order segmentation when performing the order following. If thresh is set to 0., an optimal threshold is defined for each order. Echelle keyword is THRESI. Default value is 0. A threshold can be defined for each order independently in the column :THRES of the temporary table middummr.tbl. This table is created by a preliminary execution of define/hough. After edition of the thresholds, restart with the option DEFINE/HOUGH P4=FOLLOW.

`degx,degy` degree of the 2D polynomial used to approximate the position of the orders as y = f(x,m), where
y: is the line number in pixels;
x: is the sample number in pixels;
m: a relative order number.
Echelle keyword is DEFPOL. Default is 3,4

# DEFINE/HOUGH

`hot_thres,step` ot threshold for the rejection of bright features (particle hits, hot pixels). Default is 100 000. Step for measuring orders positions when performing the order following. Default is 10.
No echelle keyword controls this parameter.

`hough_setup` This parameter is transfered as parameter P8 to the command HOUGH/ECHELLE. It enables to set the boundaries (slope limits, step) of the transformed space. See HOUGH/ECHELLE for more details.

**Note:** 1) The auxiliary table order.tbl is used as intermediate storage Parameters can be defaulted to the corresponding session values (SHOW/ECHELLE to display the current values).
2) The results will be more accurate if the keyword SCAN has been correctly set (see SCAN/ECHELLE).
3) The algorithm involves four main steps:
1) Median filtering
2) Hough transform
3) Order detection
4) Order following
Step 1 is skipped by using a minus sign (e.g. hough-par=-DENSE)
Steps 1 and 2 are skipped if hough-par=NO
Steps 1,2,3 are skipped if hough-par=FOLLOW
4) In case of emergency:
Always start with the default values (especially SCAN=0,0, hough-par=DENSE, NBORDI=WIDTHI=W
and depending on the result, consider the following:
A cause of problems in the order detection can be the presence of particle hits or defaults not removed by the median filtering. Check the intermediate image middummi.bdf. If necessary, filter the order reference frame and restart with DEFINE/HOUGH P4=-DENSE or P4=-CENTER.
The width of the orders, especially for wide orders, could be wrongly estimated. The width as estimated by the automatic program is displayed during order detection in a line like:
Detect. order N, slope xxx, interc. yyy, fwhm zzz
The real width of the orders can be measured with a command like GET/CURSOR and the session keyword WIDTHI set to this value.
If the orders present a strong curvature, the method hough-par=CENTER can be more appropriate than the default hough-par=DENSE.
Some defaults are sometimes located at the top or the bottom part of the spectrum (saturated rows, truncatured orders). To avoid them, update the keyword SCAN with the command SCAN/ECHELLE and restart with DEFINE/HOUGH.
Problems can be also due to the detection of an order badly defined, like a truncated order, which cannot be properly detected. When the procedure performs the cluster detection, check whether the slope and fwhm vary regularly. Orders are detected by order of brightness, so the intercept is not supposed to vary monotonously. If the last orders present weird parameters, it is possible to enforce the number of detected order with the command SET/ECH NBORDI=<number> and restart. If an order is wrongly detected in the middle of the cluster detection (intermediate brightness) and if this order is located in the middle of the spectrum, it wil be necessary to edit the order detection table middummr.tbl.
Before the step of order following, it is possible to edit the table middummr.tbl to correct the order detection. After a first run of DEFINE/HOUGH, check the slope, intercept, width and order number of the different orders. Restart with DEFINE/HOUGH P4=FOLLOW. If some problem occurs in the order following due to the automatic estimate of threshold, edit the table &r and update the threshold (see above parameter thresh). Restart with DEFINE/HOUGH P4=FOLLOW.
5) A description of the method is available in Ballester, 1994, Astron. Astrophys. 286, pp. 1011-1018

**Examples:** `DEFI/HOUGH stdfr`

For automatic process

`DEFI/HOUGH stdfr 12`
Enforces the detection of 12 orders

`DEFI/HOUGH stdfr 12 ? F 100.`
To redo order following (thresh=100)

## DEFINE/SKY *stdred/echelle* 07-OCT-94 PB

**Purpose:** The limits of the sky window(s) are defined directly or interactively. 1 or 2 sky windows can be defined. A cross-order profile is plotted in the graphic window and the program expects two position clicked for 1 window and four position cliked for two windows. The position of the cross-order profile plotted is set automatically (POSSKY=AUTO), indicated numerically (POSSKY=order,column) or defined interactively by a click in the display window (POSSKY=CURSOR). The cross-order profile is plotted over 1.0 order on each side (parameter half-width).

**Syntax:** `DEFINE/SKY ima [nsky] [possky] [half_width]`

| | |
|---|---|
| `ima` | Name of the original 2D echelle spectrum displayed in the display window. |
| `nsky` | Number of sky windows (1 or 2). Echelle keyword: NSKY |
| `possky` | Position of the plotted cross-order profile.<br>Echelle keyw.: POSSKY<br>Possible values are: AUTO (default), CURSOR or order,col<br>AUTO : The position corresponds to the central order<br>in the central column<br>order,col: The relative order number <order> is plotted<br>at the column <col><br>CURSOR: a cursor appears in the display windows which<br>allows to point an order. The order number and<br>column position are taken from the cursor position. |
| `half-width` | extension of the cross-order profile. Default value is 1.0 order width on each side. |

**See also:** EXTRACT/SKY

**Note:** None

**Examples:** `DEFINE/SKY rot0001 2 CURSOR 0.75`

# DISPLAY/ECHELLE

## DISPLAY/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** The command display/echelle determines the adequate scaling factors to display an image in the display window.

**Syntax:** `DISPLAY/ECHELLE image [g_flag]`

| | |
|---|---|
| `image` | Name of the image to be displayed |
| `g_flag` | (Yes/No) Optional creation f a graphic window |

**See also:** LOAD/IMAGE, GRAPH/SPEC

**Note:** None

**Examples:** `DISPLAY/ECHELLE ccd0026`

## ERROR/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** This low-level command is used in the echelle package to generate error messages associated to wrong parameter values. The name of the command in which the error occurred as well as the name and value of the offending session keyword are displayed.

**Syntax:** `ERROR/ECHELLE command keyword`

| | |
|---|---|
| `command` | Name of the command |
| `keyword` | Name of the keyword |

**See also:** VERIFY/ECHELLE

**Note:** None

**Examples:** `ERROR/ECHELLE SEARCH/ORDER SLOPE`

## EXTRACT/ECHELLE *stdred/echelle* 28-NOV-1991 PB

**Purpose:** Extract echelle orders. Position of the orders are defined in the auxiliary table order.tbl, generated by the command DEFINE/ECHELLE or DEFINE/HOUGH. The command call low level commands EXTRACT/ORDER or EXTRACT/OPTIMAL.

**Syntax:** `EXTRACT/ECHELLE input output [params] [method]`

| | |
|---|---|
| `input` | input image sampled in the usual pixel-pixel space |
| `output` | extracted image sampled in the space pixel-order |
| `params` | extraction parameters, depending on the method:<br>method LINEAR or AVERAGE: slit,offset method OPTIMAL: slit,ron,gain,sigma. |
| `slit` | slit length in pixels<br>The slit length must be slightly larger than the order to avoid periodic ripple due to the inclination of the orders. However, a slit much larger than the order will introduce interorder noise in the spectrum.<br>Echelle keyword is SLIT. |
| `offset` | offset in pixels of the center of the slit with respect to the position defined by the coefficients in the table order.tbl.<br>The offset is positive if the central line of the order is above the trace displayed by LOAD/ECHELLE.<br>Echelle keyword is OFFSET. |
| `ron,g,sigma` | read-out-noise(e-), inverse gain factor (e-/ADU), threshold (in units of the theoretical noise sigma of each pixel). See also the help file of command EXTRACT/OPTIMAL.<br>If not provided on the command line, these values are read from echelle keywords RON, GAIN, EXTSIGMA. |
| `method` | extraction method to define the way to compute the pixel values of the numerical slit as:<br>LINEAR linear interpolation (default) AVERAGE linear interpolation and average OPTIMAL optimal extraction.<br>Echelle keyword is EXTMTD. |

**Note:** The command DEFINE/ECHELLE or DEFINE/HOUGH has to be used previously.

**Examples:** `EXTRACT/ECHELLE raw extract 6`

# EXTRACT/OPTIMAL

## EXTRACT/OPTIMAL *stdred/echelle* 28-NOV-1991 MP

**Purpose:** Extract echelle orders. Position of the orders are defined in the auxiliary table order.tbl, generated with the command DEFINE/ECHELLE or DEFINE/HOUGH. Orders are extracted by calculating a weighted sum of the pixels values across the profile of the object. The algorithm is based on a paper from Koji Mukai(1990, Optimal Extraction of Cross-Dispersed Spectra, Pub. of Astr. Soc of the pacific,102:183-189).

**Syntax:** `EXTR/OPT in out slit,ord1,ord2 [ron,g,sigma] [table] [coeff]`

| | |
|---|---|
| `in` | input image sampled in the usual pixel-pixel space |
| `out` | extracted image sampled in the space pixel-order |
| `slit` | numerical slit length in pixels |
| `ord1,ord2` | number of the first and last order to extract. Relative order numbers are displayed by the command LOAD/ECHELLE. |
| `ron,g,sigma` | read-out-noise(e-), inverse gain factor (e-/ADU), threshold for the removal of cosmic ray hits (in units of the standard deviation calculated for each pixel from the number of electrons and the ron If not provided on the command line, these values are read from echelle keywords RON, GAIN, EXTSIGMA. |
| `table` | name of the input table providing the order definition. Default: order.tbl |
| `coeff` | name of the descriptor providing the coefficients of the order definition. Default: COEFF. |

**Note:** The parameter slit defines the integer length of the numerical extraction slit centered on the current order center at every position along the dispersion direction. (We suppose that the command DEFINE/ECHELLE or DEFINE/HOUGH has been used previously to find the position of the orders ) The weigths are proportional to the mean order profile perpendicular to the dispersion direction. WARNING: This version of this command is not able yet to remove cosmic rays; the command FILTER/ECHELLE should be used previously.

**Examples:** `EXTRACT/OPT raw extract 6,1,15 40,10.,3.`

## EXTRACT/ORDER *stdred/echelle* 17-AUG-1992 PB

**Purpose:** Extract echelle orders from spectra. Order positions are defined by coefficients of a regression, slit width, angle and offset. The orders are extracted by passing a numerical slit of given and angle in positions defined by the regression coefficients. The slit will sample at increments of one step in X.

**Syntax:** `EXTRACT/ORDER inp out sl,ang,off meth table coeff [ord1,ord2]`

| | |
|---|---|
| `inp` | input image in space pixel-pixel. |
| `out` | output image in space pixel-order. The orders extracted are controlled by [ord1,ord2]. |
| `sl,ang,off` | slit, angle, offset. These parameters control the width of the slit in pixels, the angle of the slit with respect to the columns of the spectrum, the offset of the slit relatively to the positions defined by the regression coefficients. |
| `meth` | extraction method. Three methods are supported: LINEAR, AVERAGE, WEIGHTED (See notes). |
| `table` | table in which the regression coefficients are stored as descriptors (command SAVE/REGRESSION). |
| `coeff` | Name of the regression coefficients as used in the command SAVE/REGRESSION. |
| `ord1,ord2` | Relative order number of the orders to be extracted. The default value 0,0 means that all orders are extracted. |

**See also:** EXTRACT/OPTIMAL, EXTRACT/ECHELLE, SAVE/REGRESSION.

**Note:** The three supported methods are the following:
LINEAR : Pixel values are added. A linear interpolation is performed.
AVERAGE : pixel values are averaged. A linear interpolation is performed.
WEIGHTED : Pixel values are weighted proportionally to the profile perpendicular to the dispersion direction.

**Examples:**
```
EXTRACT/ECHELLE ccd0001 ext1 8,0.,0.  AVERAGE order COEFF
```

# EXTRACT/SKY

## EXTRACT/SKY *stdred/echelle* 10-OCT-94 PB

**Purpose:** The limits of the sky windows must have been defined with DEFINE/SKY. An extracted sky spectrum is provided, similar to extracted echelle spectra.

**Syntax:** `EXTRACT/SKY in out [mode]`

| | |
|---|---|
| `in` | Name of input spectrum. |
| `out` | Name of output extracted spectrum. This is a 2D image in pixel-order space similar to the one produced by the command EXTRACT/ECHELLE. |
| `mode` | AVERAGE or FILTER. In mode Average, the sky windows are simply averaged. In mode Filter, each sky window is subdivided in three and the maximum pixel is rejected to eliminate particle hits. For one sky window one out of three pixel is rejected. With two sky windows, two out of six pixels are rejected. |

**See also:** DEFINE/SKY, EXTRACT/ECHELLE, DEFINE/ECHELLE, DEFINE/HOUGH

**Note:** 1) A preliminary definition of the orders with DEFINE/ECHE or DEFINE/HOUGH and of the sky windows with DEFINE/SKY is required.

**Examples:** `EXTRACT/SKY rot0002 sky2`

## FILTER/ECHELLE *stdred/echelle* 18-APR-1988 OS/DB

**Purpose:** Filter echelle spectrum for cosmic ray hits. Positions of the spectral orders must be defined in the auxiliary table order, e.g. as generated by the command DEFINE/ECHELLE. The background is also subtracted.

**Syntax:** `FILTER/ECHELLE input output`

| | |
|---|---|
| `input` | input image (normally a raw echelle spectrum or a dark subtracted spectrum) |
| `output` | filtered output image |

# FILTER/ECHELLE

**Note:** The parameters which control this command are read from echelle keywords in sections 2. (Background) and 3. (Filtering) as displayed by SHOW/ECHELLE. Parameters for the background subtraction are described in the help file of SUBTRACT/BACKGROUND. Filtering parameters are the following:

widx,widy,no_iter = width of the filter box in x and y (max. 21,21 and the number of iterations for the filtering. Recommended values: WX approximate inverse of spatial slope of orders, i.e., if for instance the orders climb by one pixel in Y every 15 pixels in X, WX should be around 15; WY somewhat larger than extraction slit length (i.e. about 9 or 11 for CASPEC), no_iter typically 3-4. defaults: 15,11,3
The corresponding echelle keyword is CRFILT.

ron,g,ethresh = read-out-noise (in e-), inverse gain factor (e-/ADU), and threshold (in units of the standard deviation) for cosmic ray removal in the spectral orders (recommended value: 4-5); defaults: 40,10,4.
The correcsponding echelle keyword is CCDFILT.

radx,rady,mthresh = radius in x and in y and threshold used for initial global median filter (same as in command FILTER/MEDIAN, see note b) below). For large radii, a high contrast between spectral orders and inter-order region can lead to unsatisfactory results (recommended values are 1,1) defaults: 1,1,10.
The corresponding echelle keyword is MEDFILT.

a) Command DEFINE/ECHELLE (or equivalent) must have been used previously in order to properly set up the table order.

b) Algorithm:
The filtering procedure as follows: First, the entire frame is filtered with a median filter (the command 'FILTER/MEDIAN input output radx,rady,mthresh NA' is executed). However, the results of this filter are retained only for the inter-order space. The regions around the spectral orders are treated separately and the results merged into frame output.
For each x-position and in all spectral orders, stripes of height widy pixels and centered on the respective order are considered. For each such stripe, the spatial profile (i.e. along the slit) is formed. The individual spatial profiles are normalized as to their integrals being unity. The 'true' local spatial profile is then formed by taking the median of all individual spatial profiles Bwithin a range widx centered on each stripe. The spatial profile of each stripe is compared with its associated local median profile. A pixel in a given stripe is rejected, if its deviation from the value of the corresponding pixel in the associated median is larger than ethresh times the expected r.m.s. error of the pixel. The expected error is computed from the characteristics of the CCD, read-out-noise ron and inverse gain factor g, and the flux in the pixel.
The algorithm is interative: If a pixel is rejected, the spatial profile is re-computed and the process is repeated, until no_iter iterations are completed.
After the global filtering by a median filter and prior to the special filtering of the spectral orders, smoothing spline polynomials of degree deg are fitted to the background and the result subtracted from the frame (command 'BACKGROUND/ECHELLE input output table hwid deg smooth' is executed). In order that subsequent high-level routines do not unnecessarily repeat this very time consuming step, a descriptor BACKGROUND is appended to the frame and initialized to 'SUBTRACTED'. The background will not be modelled again at a later stage if the descriptor BACKGROUND is found present with this and only this value.

c) This filter does not dilute or reshuffle the flux information along the dispersion axis. That is, the instrumental point spread function will be left unaltered everywhere.
d) The filter will work satisfactorily only on spectra of sources without spatial inhomogeneities along the slit. For that matter, any defects extending over the full height of an order will not be filtered out.
e) Another requirement is that the spatial slope of the orders (their tilt with respect to the x-axis) is small.
f) Because of the normalisation of the individual spatial profiles formed, it is essential to check that a proper background compensation has been obtained.

**Examples:** `FILTER/ECHELLE raw filtered`

## FLAT/ECHELLE *stdred/echelle* 04-MAR-1991 PB

**Purpose:** a) optionally subtract background from flat-field image If the descriptor BACKGROUND exists in flat-field image, the background step will be skipped. The corrected image, i.e. the flat-field corrected from its background, is stored as [correct]

b) compute an approximation to the blaze profile for the flat-field. The order-by-order blaze profiles are stored as [blaze]

The output images are used for flat-field correction.

The background is approximated by spline polynomials fitted to points located in the interorder space. These points are defined by the command DEFINE/ECHELLE Parameters can be defaulted to the current session values, as displayed by SHOW/ECHELLE.

**Syntax:** `FLAT/ECHELLE [flat] [correct] [blaze]`

| | |
|---|---|
| `flat` | flat-field image |
| `correct` | flat-field image with background subtracted |
| `blaze` | profile of the blaze function |

**Note:** Default values are read from echelle keywords in sections 2. Background and 8. Flat-Field

**Examples:** `FLAT/ECHELLE ffima ffcor blaz`

`FLAT/ECHELLE`

# HELP/ECHELLE

## HELP/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** Displays information on echelle session keyword, including type, default value, current value and description. Truncated names are accepted.

**Syntax:** `HELP/ECHELLE keyword [mode]`

| | |
|---|---|
| `keyword` | Name of the keyword. If no exact match is found, a substring case unsensitive search is performed. |
| `mode` | Search mode (K or D, default: K) indicating if the search is performed on keyword names or on description information. |

**See also:** INIT/ECHELLE, SET/ECHELLE, CLEAN/ECHELLE

**Note:** None.

**Examples:** `HELP/ECHELLE slit`

`HELP/ECHELLE wlcm`

Keyword name will be matched to WLCMTD.

`HELP/ECHELLE mtd`

Help on all keywords including the substring MTD
(e.g. EXTMTD, WLCMTD, BKGMTD, a.s.o.)

`HELP/ECHELLE ccd D`

Help on all keywords including the substring CCD
in their description (e.g. CCD, CCDFILT, RON, GAIN, a.s.o)

## HOUGH/ECHELLE *stdred/echelle* 26-MAR-1995 PB

**Purpose:** This command is a low-level command, called by DEFINE/ECHELLE to perform the automatic detection of orders on a correctly contrasted frame. The method involves Hough transform and cluster detection algorithms.

**Syntax:** `HOUGH/ECH input [scan] [stp,nbt] [nbo] [flags] [hwid] [thres] [par]`

| | |
|---|---|
| `input` | Name of order reference frame. The frame must be at standard orientation. |
| `scan` | low,high. Limits in pixels of the scanned part of the image. Pixels outside these limits in input frame do not participate to the Hough transform. |
| `stp,nbt` | step and number of traces for Hough transform. Only (nbtr) columns of the input image are transformed, separated by (step) pixels, and evenly distributed around the central column. |
| `nbo` | number of orders to be detected. If the value 0 is given, the order detection detects a minimum of 8 orders by order of brightness, then stops when no order brighter than a fraction (0.2) of the brightest order can be found. |
| `flags` | XYZ. X can be: H to perform the Hough transform;<br>N to skip it (middummh.bdf is taken as HT).<br>Y is the order detection method (see note).<br>Y can be L for low contrast method;<br>M for medium contrast method;<br>H for high contrast method;<br>N to skip the order detection.<br>Z can be V to visualize the intermediate steps;<br>N for no visualization. |
| `hwid` | Half width of the orders. A value is required if the detection method is low_contrast. If no value is provided, the half-width is measured on the frame. In medium_contrast method, the half width is measured for the first detected (or brightest) order and used for next orders. In high_contrast method, the half width is mesaured independently for each order. |
| `thres` | t1,t2,t3,t4.<br>t1,t2 are unused.<br>t3 is the hot pixel threshold. Pixels higher than t3 are ignored in Hough transform.<br>t4 is the relative threshold for the centering of detected clusters. |
| `par` | cluster,slope_mini,slope_maxi,step_slope,step_orig<br>cluster (default : 0.2) is the cluster detection constant involved in the decision of terminating the order detection. slope_mini (default : -0.1) is the minimal slope, boundary of the Hough transform image. slope_maxi (default : 0.4) is the maximal slope, boundary of the Hough transform image. step_slope (default : 0.005) is the increment of slope corresponding to 1 pixel along X axis in the Hough transformed image. step_orig (default : 1.0) is the increment of ordinate origin corresponding to 1 pixel along Y axis in the Hough transformed image. |

**Note:** For a description of the algorithm, see Ballester, 1994, Astron. Astrophys., 286, pp. 1011-1018.

**Examples:**
```
HOUGH/ECHELLE flat 5,320 10,50 0 HMV 0
```

# IDENT/ECHELLE

## IDENT/ECHELLE *stdred/echelle* 10-NOV-1994 PB

**Purpose:** Implement an instrument independent wavelength calibration for echelle spectra. Line positions are assumed to be in the auxiliary table line.tbl, as created by, e.g., SEARCH/ECHELLE.
The command works interactively in the display, assumed to be existent. The command computes dispersion coefficients if the line identification has been performed successfully.

**Syntax:** `IDENT/ECH [wlc] [lincat] [dc] [tol] [wlcloop] [wlcmtd]`

`[guess,[shift]] [ccdbin]`

`wlc` Name of wavelength calibration frame

`lincat` Name of line catalog table, which must include at least a column called :WAVE, providing laboratory wavelengths of the calibration lamp used to expose Wlc. The column :ORDER, required by command IDENT/ECH is not used anymore.

`dc` Degree of polynomial defining dispersion relation. Maximum value is 3 and also the recommended value for CASPEC.

`tol` Tolerance on rms error of global relation, Default value is 0.2.
Tol is interpreted as follows:
If Tol $>$ 0, Tol is considered in pixel units.
If Tol $<$ 0, abs(Tol) is in wavelength units.
Tol is involved at different steps of the loop:
- Standard deviation of the very first echelle relation must be smaller than Tol. Otherwise, the user is prompted whether he wants to "Start anyway".
- Tol controls the smallest wavelength window when identifying lines using a global dispersion relation.
- Before computing single-order dispersion relations, lines with residuals larger than Tol will be rejected.

Relaxing Tol value means accepting less accurate global dispersion relation in the automatic line identification and increases the risk of misiden- tifications. Values larger than 1. pixel should not be used.

`wlcloop` initial tolerance, next neighbour, maxi error. These three parameters control the loop of lines identification at the global step.
Initial tolerance (pixels) is used to check the very first dispersion relation obtained from initial identifications. If the accuracy of this starting relation is not better than the specified tolerance, the user is prompted whether it is possible to start anyway.
The second element of wlcloop controls the identification of lines by comparison of the residual of a line (computed wavelength minus position of the next catalog line) and the distance of the next neighbours in the line catalog and in the line table. If residual is less than wlcloop(2) times the distance of the next neighbour, the identification is confirmed.
Maximal error (pixels) controls the ungraceful exit of the identification loop. If the rms error of the global relation becomes larger than the specified limit (due to initial misidentifications or distortions of the wavelength calibration frame), the loop will be interrupted

`wlcmtd` Starting method. Can be PAIR, ANGLE, TWO-D, GUESS, RESTART or ORDER. Default is PAIR. See note 1.

| | |
|---|---|
| `guess` | Name of a previously saved session (cf. SAVE/ECH). The global solution of this session will be used to perform the initial line identification. Used only with method Guess.<br>An additional parameter shift can be added to the name of the guess session, separated by a coma. Its value indicates the shift in pixels between the two sets of calibration lines. If no value is provided, the shift is estimated by cross-correlation. |
| `ccdbin` | Bin factor of the CCD. This keyword is updated by the command ROTATE/ECHELLE. |

**See also:** SEARCH/ECHELLE, PLOT/RESIDUAL, PLOT/IDENT, LOAD/IDENT, CORRELATE/LINE, PLOT/CALIBRATE, LOAD/CALIBRATE

# IDENT/ECHELLE

**Note:** 1. Presentation of the methods.

The methods PAIR, ANGLE and TWO-D are interactive and use the command CENTER/MOMENT with option CURSOR to point the lines. It is recommended to beginners to read the related help and use this command before identify/echelle. The three methods can be used if the disperser is a grating. Only the method TWO-D can apply if the disperser is a grism. In case of initial misidentifications, methods PAIR and ANGLE invoke a wavelength calibration diagnosis (see Note 3).

In interactive modes, one must first point all the positions to identify, then provide order number and wavelengths.

In method PAIR, one must recognize two pairs of lines in overlapped regions of the spectrum and provide the absolute order number of the first pointed line and wavelengths for each of the two pairs. The geometry of the pairs provide the angle of orientation of the spectrum.

In method ANGLE, one identify at least four lines anywhere and provide absolute order number of the first pointed line and the wavelengths of all lines. The angle of orientation of the spectrum is computed analytically.

At the difference of PAIR and ANGLE which involve the echelle relation (product order by wavelength as a function of the x position), the method TWO-D fits from the beginning a bivariate polynomial to the identifications. Therefore, it requires more initial identifications than the above methods. The minimum number is given by (dc + 1)**2 + 1, where dc is the degree of the polynomial used to fit the dispersion relation.

Method RESTART must be used only IMMEDIATELY AFTER the methods PAIR, ANGLE or TWO-D. It avoids the interactive pointing of the lines and enters the program at the step of the identifications.

Method GUESS is not interactive and can be used after any calibration by an interactive method. The method GUESS requires the name of a previously saved echelle session, which process tables will be used to start automatically the lines identification. The shift between the two sets of calibration lines is estimated by cross-correlation, unless its value is provided (see parameter guess).

Method ORDER allows to restart only the single-order solutions with a different parameter wlctol.

2. Input parameters

Echelle keywords are shown in section 6. Wavelength calibration of the SHOW/ECHELLE).If not provided, parameters are read from echelle keywords. One additional parameter is only read from the echelle session:
- number of orders (fixed by DEFINE/ECH).

3. Initial identifications, Error diagnosis

The command first try to fit an accurate global dispersion relation from the minimum four identifications provided by the user (either two times two lines in overlapped regions of successive orders (PAIR), or four lines anywhere (ANGLE).

If it appears that the initial relation is not accurate enough, the user will be prompted for corrective actions, which can consist of:
- starting anyway
- perform initial identifications again.

**Note:** In methods PAIR and ANGLE, a wavelength calibration diagnosis is invoked. The diagnosis makes use of the redundancy in the input data to check whether one of the values is possibly wrong. Therefore, no more than one error can be recovered. By solving equations, all sets of (n-1) parameters are used to compute the optimal n-th parameter and the resulting rms obtained by replacement of the actual value by the optimal value is computed. The interpretation of the diagnosis is normally done as follows: A linear fit of the echelle relation with the original values is computed. Read the rms of this fit and take it as reference value. The optimum for each parameter is computed. A diagnosis is possible if one of the values improves dramatically the rms. Then use the mathematical optimum as an indication of the correct value. A very common mistake in the initial identifications is the order number wrong by one.

4. Identification loop

The initial solution being accepted, an iterative loop improves the global solution by identifying more lines. The convergence is achieved when no new lines can be identified. Finally, order by order solutions are either fitted, or estimated from the global solution, depending on the number of lines available for each order. Residuals are computed for every order (of course unless there is no identified line in that order).

5. Final solution

All fits are performed using polynomials. Calibration results are written in the table line.tbl and will be used in next reduction steps or as a guess for another calibration.

The command IDENTIFY/ECHELLE accepts an additional option controlled by the session keyword WLCOPT, which value can be 1D or 2D. This keyword controls the method used to compute the coefficients of the single order coefficients: in 1D mode the coefficients are computed independently for each order, in 2D mode the coefficients are derived from a global 2D solution. The default value of WLCOPT is 1D.

**Examples:**
```
IDENT/ECHELLE mywlc mycat 2 0.25 3.0 ANGLE

IDENT/ECHELLE P6=GUESS P7=oldses,-1.

IDENT/ECH
```

---

## INIT/ECHELLE *stdred/echelle* 17-DEC-1991 PB

**Purpose:** Initializes the session parameters for the echelle reduction. The command works in two modes:
- Defaulting the name, all the parameters are set to the default value.
- If the session name is given, parameters and working tables are taken from that session.

**Syntax:** `INIT/ECHELLE [name]`

`name` optional name with the session name

**Note:** The command "SAVE/ECHELLE name" has to be used previously in order to save the session.

**Examples:** none

# INITIAL/EMMI

## INITIAL/EMMI *stdred/echelle* 09-MAR-95 PB

**Purpose:** The command INIT/EMMI reads header information from EMMI Echelle images and initialises the Echelle context to the corresponding configuration. The configuration can also be set directly.

**Syntax:** `INITIAL/EMMI [ref] [grism]`

`ref` If no value is provided for this parameter, the displayed image (if it exists) will be used as reference. If an image name is provided, the header information will be read from this image. If a number is provided it will be interpreted as the grating number (9 or 10) and the second parameter is expected.

`grism` If a numerical value has been provided as first parameter (grating number), the second parameter must indicate the grism number (3 or 4 for grating 9, 3 to 6 for grating 10).

**See also:** UPDATE/ORDER, UPDATE/ECHELLE.

**Note:** 1) Pre-calibrated EMMI solutions are stored in the MID_EMMI directory. These solutions are defined in world coordinates and apply to EMMI images flipped with the command FLIP/IMAGE to be set at standard orientation. The command INIT/EMMI will attempt to read the descriptors _EIG1_ID and _EIO9_ID from the image header and will activate the command:
INIT/ECHELLE MID_EMMI:echGRgCD where GR and CD stand for grating and cross-disperser numbers.
2) The pre-calibrated solutions apply to images at standard orientation, namely EMMI images flipped with the command FLIP/IMAGE.
3) The solution can be visualized on an image using LOAD/ECHELLE. An Offset between the displayed image and the pre-calibrated solution is usually present and must be corrected with the command UPDATE/ORDER.
4) The MID_EMMI directory must be installed in the /midas/calib area. The calibration tables are distributed on request in complement to the Midas releases. These tables are also available on anonymous ftp at the host ftphost.hq.eso.org (IP number 134.171.40.2). The files to be retrieved are located in the directory /midaspub/calib and are named README.calib and calib.tar.Z. Command SHOW/TABLE can be used to visualize the column name and physical units of the tables. Demonstration data required to execute the tutorial procedure TUTORIAL/ECHELLE are also located on this ftp server in the directory /midaspub/demo as echelle.tar.Z. FTP access is also provided on the World Wide Web URL:
http://http.hq.eso.org/midas-info/midas.html
The calibration directory contains other information such as characteristic curves for ESO filters and CCD detectors, which can be visualized with the Graphical User Interface XFilter (command CREATE/GUI FILTER).
5) ThAr calibration tables are located in MID_ARC directory. A standard ThAr table is provided as thar.tbl and can be copied with the system command:
copy MID_ARC:thar.tbl thar.tbl (VMS)
cp $MID_ARC/thar.tbl thar.tbl (UNIX)
Other ThAr tables selected for blends at different resolutions have been provided by H.Hensberge & al. (Antwerp University) for the spectral resolutions 25000, 33000, 50000 and 100000 and are named thar25.tbl, thar33.tbl, thar50.tbl and thar100.tbl.
6) Spectrophotometric standards are provided in the MID_STANDARD directory, usually corresponding to the directory /midas/calib/data/spec/flux. Additional tables are available in /midas/calib/data/spec/ctio.

**Examples:** `INITIAL/EMMI`
Reads header information from the image currently present in the display window and initializes the context.

`INITIAL/EMMI ccd0026`
Reads header information from the image ccd0026.bdf and initializes the context.

`INITIAL/EMMI 10 4`
Retrieves pre-calibrated solution for grating 10, cross-disperser 4 by issuing the command: INIT/ECH MID_EMMI:ech10g4.

## KEYDEL/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** This low-level command is involved in context cleaning and allows to delete the echelle session keyword. The corresponding descriptors can also be removed from a table.

**Syntax:** `KEYDEL/ECHELLE [table]`

| | |
|---|---|
| `table` | Name of the table from which the decsriptors will be removed. |

**See also:** REGISTER/SESSION, CLEAN/ECHELLE

**Note:** None

**Examples:** `KEYDEL/ECHELLE`

# LOAD/CALIBRATION

## LOAD/CALIBRATION *stdred/echelle* 10-AUG-1995 PB

**Purpose:** Displays the extracted and rebinned wavelength calibration frame for verification of the regularity of the order by order dispersion relations.

**Syntax:** `LOAD/CALIBRATION [input] [output]`

| | |
|---|---|
| `input` | Input image. By default the wavelength calibration frame (Session keyword: WLC). |
| `output` | Output image. Default: middummr.bdf |

**Note:** 1) Display cuts are copied from the input image. The output image can be reloaded with new cuts for a better visualisation.
2) The wavelength range should regularly increase from bottom to top on the displayed image.
3) If necessary the regularity of the dispersion relations can be improved by using the option WLCOPT=2D fitting a bivariate polynomial to the dispersion relation. Once a satisfactory calibration has been obtained with the method WLCMTD=PAIR, ANGLE or GUESS and WLCOPT=1D, new coefficients for the 2D solution are given with the sequence:
SET/ECHELLE WLCOPT=2D WLCMTD=ORDER
IDENT/ECHELLE

4) Method WLCMTD=ORDER can also be used to recompute the dispersion coefficients for another tolerance value (Session keyword TOL).

**See also:** LOAD/IDENT, PLOT/RESIDUAL, IDENT/ECHELLE, REBIN/ECHELLE, LOAD/CALIB, PLOT/CALIB, PLOT/IDENT

**Examples:** `LOAD/CALIB`

## LOAD/ECHELLE *stdred/echelle* 20-AUG-1991 PB

**Purpose:** Display on the overlay plane of the display the positions of the echelle orders. Order positions are stored in the auxiliary table order.tbl.

**Syntax:** `LOAD/ECHELLE`

**Note:** The corresponding image has to be displayed on the monitor

Descriptors START and STEP of the corresponding image must preferably be set both to 1.,1.

The display of background positions is controlled by the echelle keyword BKGVISU. Possible values are YES or NO. Default is YES.

**Examples:** `none`

## LOAD/IDENTIFICATION *stdred/echelle* 22-AUG-1991 PB

**Purpose:** Display on the overlay plane of the monitor the positions of the identified lines. Line positions are stored in the auxiliary table line.tbl

**Syntax:** `LOAD/IDENTIFICATION`

**See also:** IDENTIFY/ECHELLE, PLOT/RESIDUAL, LOAD/CALIBRATE, PLOT/CALIB, PLOT/IDENT

**Note:** The corresponding image has to be displayed on the monitor

**Examples:** `none`

## LOAD/SEARCH *stdred/echelle* 09-MAR-95 PB

**Purpose:** Loads on display the position of the lines found by the SEARCH/ECHELLE command.

**Syntax:** `LOAD/SEARCH`

**See also:** SEARCH/ECHELLE

**Note:** None

**Examples:** `LOAD/SEARCH`

# MERGE/ECHELLE

## MERGE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Produces a 1-D spectrum, from the order by order file generated by REBIN/ECHELLE. Two methods are available: NOAPPEND individual orders are separated in 1D files AVERAGE the orders are merged into a 1D file, the algorithm computes a weighted average in the overlapping region of adjacent orders. The normalized weight is a linear ramp between 0 and 1 in the overlapping region.

**Syntax:** `MERGE/ECHELLE inframe outframe [params] [method]`

| | |
|---|---|
| `inframe` | input image, sampled in the space wavelength-order Corresponds to a extracted and rebinned spectrum. |
| `outframe` | one dimensional spectrum, sampled in wavelengths. Values in the overlapped region are computed according to the method |
| `params` | method NOAPPEND : order numbers (ord1,ord2). method AVERAGE : wavelength interval to be skipped at both edges of the overlapping region. Default is 3 Angstroem. Echelle keywords are DELTA (if MRGMTD=AVERAGE) or MRGORD (if MRGMTD=NOAPPEND). |
| `method` | optional parameter to define the overlapping region of the orders as: NOAPPEND : get individual orders as 1D files. AVERAGE : merge all the orders into a 1D file. Default method. Echelle keyword is MRGMTD. Default is AVERAGE |

**Note:** none

**Examples:** `MERGE/ECHELLE extr1 spec1 3. AVERAGE`

## MERGE/OPTIMAL *stdred/echelle* 09-MAR-95 PB

**Purpose:** Optimal weighted merging of echelle orders. A weight image is required.

**Syntax:** `MERGE/OPTIMAL rebima weight out [delta]`

| | |
|---|---|
| `rebima` | Input rebined echelle spectrum, as provided by REBIN/ECHELLE. |
| `weight` | Standard deviation image. The actual coefficients involve in the weighted merging will correspond to the variance (squared standard deviation image). |

**See also:** MERGE/SPECTRUM, MERGE/ECHELLE

**Note:** None

**Examples:** `MERGE/OPTIMAL reb26 ww26 sp26`

## OFFSET/ECHELLE *stdred/echelle* 17-MAR-95 PB

**Purpose:** In reason of different positioning along the slit, echelle observations might present an offset with respect to the image used to define the order positions. The command OFFSET/ECHELLE determines this offset by performing a number of offset measurement at random positions on a raw spectrum.

**Syntax:** `OFFSET/ECHELLE [image] [range] [cover] [ordtab] [mode]`

| | |
|---|---|
| `image` | Image name (echelle spectrum at standard orientation). By default the image in the display window is used for offset measurement. |
| `range` | Range of search for the offset (default 20 pixels) |
| `cover` | Number of measurement points (default 30) |
| `ordtab` | Name of the order table (Default: sess. keyw. ORDTAB) |
| `mode` | Mode of action (Verb, Silent). Default mode is Verbose in which the OFFSET keyword is set to the measured offset value (the OFFSET keyword is used by the EXTRACT/ECHELLE command). In mode Silent the offset value is stored in OUTPUTR(1) and the OFFSET keyword is left unchanged. |

**See also:** DEFINE/ECHELLE, DEFINE/HOUGH, EXTRACT/ECHELE, LOAD/ECHELLE

**Note:** 1) The algorithm involves the measurement of the offset at a number of positions selected randomly over the frame. Although the algorithm is robust to particle hits, it is possible to filter the image before processing. The signal to noise of the spectrum should be sufficient to allow the unambiguous detection of the central peak of the orders.

**Examples:**
```
OFFSET/ECHELLE value1
Comment line

OFFSET/ECHELLE value1 value2
Comment line
```

# OVERLAP/ECHELLE

## OVERLAP/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** Plots the overlaps between adjacent orders of a rebinned echelle spectrum.

**Syntax:** `OVERLAP/ECHELLE rebima order`

| | |
|---|---|
| `rebima` | Name of the rebinned echelle spectrum, as produced by REBIN/ECHELLE. |
| `order` | Relative order number. The overlap between this order and the next will be plotted. |

**See also:** REBIN/ECHELLE, PLOT/SPECTRUM.

**Note:** None

**Examples:** `OVERLAP/ECHELLE &w 20`

## PLOT/CALIBRATE *stdred/echelle* 28-AUG-1995 PB

**Purpose:** Plot the pixel bin size as a function of the wavelength for the different orders. The auxiliary table line.tbl is needed by this command.

**Syntax:** `PLOT/CALIBRATE [ord1,ord2]`

| | |
|---|---|
| `ord1,ord2` | optional screen order range. If omitted the complete order range is plotted. |

**See also:** IDENTIFY/ECHELLE, PLOT/RESIDUAL, LOAD/CALIBRATE, LOAD/IDENT

**Note:** The command is used to control the dispersion coefficients at the end of the wavelength calibration step

**Examples:** `PLOT/RESIDUAL`

## PLOT/ECHELLE *stdred/echelle* 09-MAR-1995 PB

**Purpose:** Plot extracted orders of echelle images in the space pixel-order. The command EXTRACT/ECHELLE for the corresponding image has to be used previously.

**Syntax:** `PLOT/ECHELLE frame [ord1,ord2] [printer]`

| | |
|---|---|
| `frame` | input image, in the space pixel-order |
| `ord1,ord2` | start and end relative order number. |
| `printer` | optional printer name. If a name is specified, plots are sent to this printer. |

**See also:** PLOT/SPECTRUM, PLOT/IDENT

**Note:** none

**Examples:**

```
PLOT/ECHELLE extima 1,20
PLOT/ECHELLE extwlc 1,16 ps2usr1
```

## PLOT/IDENTIFICATION *stdred/echelle* 22-AUG-1991 PB

**Purpose:** Plot line identifications. The extracted image in the space pixel-order is plotted on the graphic window order by order and the corresponding identified lines stored in the auxiliary table line.tbl are displayed. The command is used to control the wavelength calibration step

**Syntax:** `PLOT/IDENTIFICATION frame [ord1,ord2] [printer]`

| | |
|---|---|
| `frame` | input image, in the space pixel-order |
| `ord1,ord2` | relative number of start and end order. Def:1,1 |
| `printer` | optional printer name. If a name is specified, plots are sent to this printer. |

**Note:** Line identifications are stored in the auxiliary table line.tbl

**Examples:**

```
PLOT/IDENTIFICATION extima 1,20 ps2usr1
PLOT/IDENT extwlc 1,17
```

# PLOT/RESIDUALS

## PLOT/RESIDUALS *stdred/echelle* 28-AUG-1995 PB

**Purpose:** Plot the residuals of the wavelength fit in the echelle reduction The auxiliary table LINE is used by this command Residuals are plotted order by order if the parameters are specified.

**Syntax:** `PLOT/RESIDUAL [ord1,ord2]`

`ord1,ord2` optional screen order range. If omitted the complete order range is plotted.

**See also:** IDENTIFY/ECHELLE, LOAD/CALIBRATE, LOAD/IDENT, PLOT/CALIBRATE

**Note:** The command is used to control the dispersion coefficients at the end of the wavelength calibration step

**Examples:** `PLOT/RESIDUAL`

## PLOT/SPECTRUM *stdred/echelle* 07-OCT-94 PB

**Purpose:** After extraction (extract/echelle) and resampling (rebin/echelle) a spectrum can be plotted in wavelength range.

**Syntax:** `PLOT/SPECTRUM in [start,end]`

`in` name of the rebinned spectrum.

`start,end` wavelength range of the plot. By default the graphic window set-up is taken into account.

**See also:** PLOT/ECHELLE, PLOT/IDENT, EXTRACT/ECHELLE, REBIN/ECHELLE

**Note:** 1) The y-axis limits of the plot are by default determined by the range of the first order plotted. It is usually preferable to set the limits using set/graph yaxis=...

**Examples:** `PLOT/SPECTRUM reb24 4500,5500`

## PREPARE/BACKGROUND *stdred/echelle* 10-AUG-1994 PB

**Purpose:** Low level command called by BACKGR/ECHELLE. Prepare table of background reference positions from order fitting coefficients stored in table [order_tab] and descriptor [descr].

**Syntax:** `PREPARE/BACKGROUND [step] [init] [bkgtab] [ordtab] [descr]`

| | |
|---|---|
| `step` | step along X axis between background reference positions. |
| `init` | initialization flag to enforce the creation of a new table (init = INI). Default is NULL: a new table will be created only if the step or scan parameters have been modified, or if the background table does not exist. |
| `bkgtab` | name of output background table. Default is back.tbl. |
| `ordtab` | name of input order table. Default is order.tbl. |
| `descr` | generic name of descriptors conataining coefficients of orders fitting. Default is COEFF. |

**Examples:** `PREP/BACK 5 INI`

## PREPARE/WINDOW *stdred/echelle* 30-JUL-1991 PB

**Purpose:** Write the descriptors FLAT_BKG and LHCUTS onto the images in the catalogue. These descriptors are used by the command AVERAGE/WINDOW in order to remove bad pixels from the raw data. Specially recommended for dark exposures to produce the BIAS and pre-flashed BIAS frames. Additionally, the command check the presence of the descriptor with the exposure time and will prompt for it if not present. The command works on catalogues containing the input images. The catalogue is created with CREATE/CATALOG catalogue dirspec (See HELP)

**Syntax:** `PREPARE/WINDOW catalogue flatbkg lhcuts`

| | |
|---|---|
| `catalogue` | name of the catalogue with the input images |
| `flatbkg` | value to be written into descriptor FLAT_BKG |
| `lhcuts` | values to be written into descriptor LHCUTS |
| | See AVERAGE/WINDOW for more information about these two parameters. |

**Note:** none

**Examples:** `AVERAGE/WINDOW mycat 3. 20,50.`

# REBIN/ECHELLE

## REBIN/ECHELLE *stdred/echelle* 20-OCT-1992 PB

**Purpose:** Rebin extracted orders into linear wavelength steps. The frame is converted from the space pixel-order into the space wavelength-order

**Syntax:** `REBIN/ECHELLE input output sample`

| | |
|---|---|
| `input` | input image, sampled in the space pixel-order Corresponds to the extracted orders |
| `output` | output image, sampled in the space wavelength-order |
| `sample` | output sampling domain, given as a real number defining the sampling step in Angstroms or as a filename of the reference image used to define start and step for each order<br><br>Echelle keyword is SAMPLE. |

**Note:** The rebinned spectrum is a two-dimensional image in the space wavelength-order. Therefore, the descriptor START of this image cannot indicate the starting wavelength, which is different for each order. To produce a final spectrum including correct descriptors START and STEP, use the command MERGE/ECHELLE.

**Examples:** `REBIN/ECHELLE extord1 extsp1 0.2`

## REDUCE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Object reduction of echelle data. There are two available methods (parameter REDUCTION): STANDARD consists of the following steps
. Background correction
. Flat field correction (optional, depending on FFOPT)
. Order extraction
. Calibration in wavelengths
. Response correction (optional, depending on RESPOPT)
. Order merging
The parameters involved in this command are displayed by SHOW/ECHELLE. Before using the command, the following sections must be checked:

2. Background 4. Extraction 7. Rebin 8. Flat field correction 9. Response correction 10. Merging

**Syntax:** `REDUCE/ECHELLE input output [bkcor]`

| | |
|---|---|
| `input` | raw image |
| `output` | 1D reduced spectrum |
| `bkcor` | if an image name is provided, a simplified reduction is performed, including background correction, extraction, rebin, and theoptional flat-field correction. |

**Note:**

**Examples:** `REDUCE/ECHELLE raw spectrum`

## REGRESSION/ECHELLE *stdred/echelle* 25-MAR-1992 PB

**Purpose:** Fit a 2D polynomial to the order positions. The command is used internally by DEFINE/ECHELLE and DEFINE/HOUGH. An iterative rejection of outliers is performed by kappa-sigma clipping.

**Syntax:** `REGRESSION/ECHELLE [defpol] [niter] [absres] [kappa]`

| | |
|---|---|
| `defpol` | degree of the 2D polynomial. Default is 3,4. Check the current value with the command SHOW/ECHELLE. Echelle keyword is DEFPOL. Limit is 5,5. |
| `niter` | number of additional loops for outlier rejection. No related echelle keyword. Default is 3. |
| `absres` | maximum allowed residual in pixels. Default is 2.0 No related echelle keyword. |
| `kappa` | number of standard deviation. Default is 4.5 No related echelle keyword. |

**Note:** The algorithm involves the following steps:
- The rms error to a linear fit is estimated for each order and written to a temporary table middummw.tbl
- A maximum admissible rms is computed from the median of the order by order rms's. maxrms = 3.5 * median(rms(i)), i=1,nb of orders
- All orders which rms is larger than maxrms are rejected from the first bivariate polynomial fitting.
- An iterative polynomial fitting is performed, rejecting outliers by a kappa-sigma clipping.

**Examples:** `REGRESSION/ECHELLE 3,4`

## REPEAT/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Modify deviating pixels in the response file interactively. This command has to be used if the response file is wrong due to the presence of spectral features in the standard star (due to the low resolution of the available fluxes). Parameters can be defaulted to the current session values, as displayed by SHOW/ECHELLE.

**Syntax:** `REPEAT/ECHELLE [scalx,scaly] [response]`

| | |
|---|---|
| `scalx,scaly` | scaling factors to display the response file scaly cannot be negative. |
| `response` | instrument response Echelle keyword is RESPONSE. |

**Note:** The command will display the 2D response and will prompt first to check the cuts of the displayed image, then to remove bad pixels in the response using the cursor.

Performed corrections can be acknowledged or rejected in the end.

**Examples:** `REPEAT/ECHELLE -2,10`

# RESPONSE/ECHELLE

## RESPONSE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Compute instrument response by comparing the extracted orders of the standard star to a table of fluxes. Optional step in the STANDARD reduction of echelle spectra. Parameters can be defaulted to the current session values, as displayed by SHOW/ECHELLE.

**Syntax:** `RESPONSE/ECHELLE [std] [fluxtab] [response]`

| | |
|---|---|
| `std` | standard star raw image Echelle keyword is STD |
| `fluxtab` | is the table containing flux reference Echelle keyword is FLUXTAB |
| `response` | instrument response Echelle keyword is RESPONSE |

**Note:** a) In addition to parameters STD, FLUXTAB, RESPONSE, the command uses echelle keywords FILTMED, FILTSMO, PIXNUL (see below). Keywords FILTMED and FILTSMO correspond to parameter [filter_specs] of MIDAS commands FILTER/MEDIAN and FILTER/SMOOTH

b) The background will not be computed if descriptor BACKGROUND/C/1/12 exists in the frame <std>. The background is displayed in a graphic window if BKGVISU=YES. Set it to NO to avoid the display.

c) The response function is computed the following way:

- The background is subtracted from standard star spectrum - An optional flat-fielding is performed (keyword FFOPT) - Standard star spectrum is extracted and rebinned. - Standard star spectrum is filtered, first by a median filter, then by a smoothing filter (keywords FILTMED, FILTSMO). - An image is created from the flux table at the same start and step than the standard star spectrum - response is the flux reference divided by the standard star spectrum. - left and right parts of the response function are set to zero (keyword PIXNUL).

d) See also Section 9. Response of SHOW/ECH

**Examples:** `RESPONSE/ECHELLE feige56.bdf feige56.tbl response`

## RIPPLE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Correct for the echelle blaze function. Input data results from the sequence of commands
EXTRACT/ECHELLE
REBIN/ECHELLE
Three methods are implemented. The first one (SINC) is based on Ahmad, 1981, NASA IUE Newsletter, 14, 129. The second method (OVER) is based on BARKER, 1984, Astron.J., 89, 899. In this case only the parameter k is fitted. The third method (FIT) is an extension of OVER to fit both parameters k and alpha. See manual for detailed command description.

**Syntax:** `RIPPLE/ECHELLE input output [params] [method] [option]`

| | |
|---|---|
| `input` | extracted orders, sampled in wavelengths. Produced by EXTRACT/ECHELLE and REBIN/ECHELLE. |
| `output` | ripple corrected orders. Use the command MERGE/ECHELLE to get 1D files. |
| `params` | parameters. They depend on the method.<br>Method SINC: k,alpha<br>Method OVER: lambda1,lambda2,k,alpha<br>Method FIT: lambda1,lambda2,k,alpha<br>See manual for detailed command description. The coresponding echelle keywords are LAMBDA1, LAMBDA2, RIPK, ALPHA. |
| `method` | defines the method used for correction. Possible methods are SINC, OVER or FIT. Echelle keyword is RIPMTD. |
| `option` | Allows to freeze the values of K and Alpha during the computation. Possible options are NO or FREEZE. Default is NO. Echelle keyword is RIPFRZ. |

**Note:** Recommended values (for Caspec) for 31.6 grooves/mm resp. 52 grooves/mm:

| | | | |
|---|---|---|---|
| k | 568746.0 | 344705.0 | |
| alpha | 0.8 | 0.8 | |
| lambda1 | 1. | 1. | |
| lambda2 | 20. | 20. | |

**Examples:** `RIPPLE/ECHELLE input output 568746.,0.8`

# ROTATE/ECHELLE

## ROTATE/ECHELLE *stdred/echelle* 12-OCT-1994 PB

**Purpose:** Rotate and optionally flip echelle images. The output result has to be in such a way that wavelengths increase from left to right and spectral order numbers increase from top to bottom. All possible combination of rotation and flip are made available by the parameters angle and flip_axis.

Additionally, the command checks the presence of the descriptor with the exposure time and will prompt for it if not present.

Descriptors START and STEP of rotated images are set to 1.,1. STEP descriptor of original images correspond to the bin factor and are written in echelle keyword CCDBIN.

The command works on catalogues containing the input images. The catalogue is created with CREATE/CATALOG catalogue dirspec (See HELP)

**Syntax:** `ROTATE/ECH cat,ima root [mode] [flip_axis] [angle] [o_time]`

| | |
|---|---|
| `cat,ima` | name of the catalogue with the input images. This parameter will be interpreted as a catalog name by default (no extension given or extension .cat). Other extensions (e.g. .bdf) will be interpreted as single image names. |
| `root` | root of the name of the resulting images (up to 4 alphanumeric characters starting with a letter). If a single image name is provided as input, root is the name of the output image. |
| `mode` | optional parameter to define the mode of operation: ROTATE - rotation only. The angle of rotation is defined by parameter angle.<br>FLIP - rotate and flip, (default) Angle of rotation and flip axis are defined by angle and flip_axis. |
| `flip_axis` | Flip Axis. Used only if mode = FLIP. Possible values: X,Y,XY Default: X. See command FLIP/IMAGE for more information |
| `angle` | Angle of rotation of the image, in degrees. Default: 90. degrees. |
| `o_time` | Default observation time. The procedure checks for each image that the descriptor O_TIME/D/7/1 exists and is set to a non null value. If not the default value is taken into account. If no default value is provided, the user is prompted for the observation time. |

**Note:** none

**Examples:** `ROTATE/ECHELLE mycat a rota`
The images in mycat are rotated. You will find the output as a0001, a0002, ...

## SAVE/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Saves the current parameters and auxiliary tables order.tbl, line.tbl and back.tbl.

**Syntax:** `SAVE/ECHELLE name`

| | |
|---|---|
| `name` | session name. |

**Note:** The command "INIT/ECHELLE name" can be used to restore the session status.

**Examples:** `none`

## SAVINIT/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** Low-level command allowing to save or retrieve the echelle session keyword from an image or table. The keywords are saved as descriptors of identical name in the Midas structure. This command is used in SAVE/ECHELLE and INIT/ECHELLE to save and initialize echelle sessions. The command can also be used to save the session parameters as descriptors of a reduced observation.

**Syntax:** `SAVINIT/ECHELLE ima,tab mode`

| | |
|---|---|
| `ima,tab` | image or table name, including extension (.tbl or .bdf) |
| `mode` | READ or WRITE to read (write) the descriptors from (to) the data structure. |

**See also:** REGISTER/SESSION, SAVE/ECHELLE, INIT/ECHELLE

**Note:** None

**Examples:** `SAVINIT/ECHELLE order.tbl WRITE`

## SCAN/ECHELLE *stdred/echelle* 30-JUL-1991 PB

**Purpose:** Allows to update keyword SCAN by different options (direct input of values, defaults, interactive). Parameter SCAN is used by many commands to ignore outside parts of the spectra (order definition, background). It is normally set when starting an echelle session. The keyword SCAN is saved by the command SAVE/ECHELLE.

**Syntax:** `SCAN/ECHELLE frame [scan-par]`

| | |
|---|---|
| `frame` | input image, in the space pixel-pixel |
| `scan-par` | scan parameters. Default is NOSCAN. |
| | Can be: start,end : limits of the scanned part of the image, in pixel |
| | NOSCAN : take 1 and number of rows as default values. |
| | CURSOR : optionally creates a graphic window and prompt for interactive setting. |

**Note:** none

**Examples:** `SCAN/ECHELLE ccd0008 5,320`

`SCAN/ECHELE ccd0008 CURSOR`

# SEARCH/ECHELLE

## SEARCH/ECHELLE *stdred/echelle* 18-DEC-1986 JDP

**Purpose:** search for calibration lines in a wavelength calibration echelle frame in the space pixel-order. The positions the detected lines are stored in the auxiliary table LINE. The command uses internally SEARCH/LINE with the method GAUSSIAN (default) or GRAVITY on EMISSION lines (See the parameter SEAMTD of the command SET/ECHELLE).

**Syntax:** `SEARCH/ECHELLE frame [width2,thres2]`

| | |
|---|---|
| `frame` | input image corresponding to a extracted wavelength calibration frame in the space pixel-order, as produced by the command EXTRACT/ECHELLE |
| `widht2` | is the approx. width of the lines along the dispersion direction in pixels. Echelle keyword is WIDTH2. Default is 5. |
| `thres2` | is the detecting threshold relative to the local background. Echelle keyword is THRES2. Default is 50. |

**Note:** The command EXTRACT/ECHELLE has to be used previously

**Examples:** `SEARCH/ECHELLE extwlc 6,100`

## SEARCH/ORDER *stdred/echelle* 30-JUL-1991 PB

**Purpose:** Low-level command called by DEFINE/ECHELLE. Search for order positions in a 2-D echelle frame. The positions of points located in order and in interorder space are stored in auxiliary table ordtab.

**Syntax:** `SEARCH/ORDER [ordref] [width1,thres1,slope] [ordtab] [defmtd]`

| | |
|---|---|
| `ordref` | raw input image. Must be at standard orientation. Echelle keyword is ORDREF. |
| `width1` | is the full width at half maximum of the orders accross the dispersion direction (in pixels). Default is 8. Echelle keyword is WIDTH1. |
| `thres1` | is the absolute threshold for the order detection Default is 1000. Echelle keyword is THRES1 |
| `slope` | is the mean slope of the orders. Default is 0.05 Echelle keyword is SLOPE |
| `ordtab` | is the name of ouput table. Default: order.tbl |
| `defmtd` | defines the method used to detect the orders. Possible methods are : STD: the standard (default) method, recommended for well separated orders (e.g., for CASPEC above 4700 A). COM: thresholding is done after complementing pixel values. Useful where orders are more crowded (interorder space less than order width). Echelle keyword is DEFMTD. |

**Note:** The auxiliary table order is used as intermediate storage Parameters can be defaulted to the corresponding session values (SHOW/ECHELLE to display the current values).

The performance of the algorithm depends sensitively on the value entered for the width of the order profile.

In its present version, the algorithm will crash if the exposure level of a detected order drops below the detection threshold. This can cause problems if, as can be the case with the Echellec, in the center of the frame the scattered light level in the interorder region is higher than the exposure level within the orders towards their end(s). Use **EXTRACT/IMAGE** to extract the useful part of the frame.

**Examples:** `SEARCH/ORDER stdframe 6,10`

## SELECT/BACKGROUND *stdred/echelle* 03-MAR-1991 PB

**Purpose:** Enables to unselect interactively reference positions from table ORDER before background estimate. The background will be interpolated between selected positions.

**Syntax:** `SELECT/BACKGROUND [all]`

| | |
|---|---|
| `all` | Enables to start with all positions selected (all=ALL) Default is NULL: start with last selected positions. |

**Note:** none

**Examples:**

```
SELECT/BACKGROUND ALL
SELECT/BACKGROUND
```

# SET/ECHELLE

## SET/ECHELLE *stdred/echelle* 16-JUN-1992 PB

**Purpose:** Define the values of parameters in the current session.

**Syntax:** `SET/ECHELLE par=value [...]`

`par` parameter name

`value` parameter value

**Note:** a) Space characters cannot be inserted on the right or left part of the sign "=". The name of the echelle keyword can be truncated to a non ambiguous root. Number of parameter assigned is limited to 8.

b) String values can be specified between double quotes. The double quote (") is a reserved character and cannot be used in a value.

c) Short description of echelle keywords is provided by HELP/ECHELLE command, values are displayed by SHOW/ECHELLE.

d) It is possible to assign a value to echelle keywords using monitor assignment. In this case, there must be a space on both parts of the sign =. The syntax depends on the definition of the keyword (provided by HELP/ECHELLE). Examples: ORDREF = "ff456.bdf" (keyw. ORDREF/C/1/60) WIDTH1 = 12 (keyw. WIDTH1/I/1/1) MEDFILT(2) = 4. (keyw. MEDFILT/R/1/3) To assign values to character keywords, additional spaces must be inserted to overwrite a longer previous value.

e) To assign the value of an element of a multiple element keyword (e.g. MEDFILT/R/1/3) it is possible to use the direct assignment (MEDFILT(2) = 4.) or the SET/ECH by inserting comas in place of the values that must not be changed.

f) The total length of the line is limited to 132 characters.

**Examples:** `SET/ECH ORDR=ff456.bdf WIDTH1=12 MEDFI=,4 INSTR="NTT EMMI"`

## SHOW/ECHELLE *stdred/echelle* 25-SEP-1991 PB

**Purpose:** Show the parameters of the current session.

**Syntax:** `SHOW/ECHELLE`

**Note:** 1) The parameters displayed by SHOW/ECHELLE depend on the values of method and option keywords. The list of these keywords and a short description is provided by commands "HELP/ECH method" and "HELP/ECH options"

The displayed parameters correspond to the only parameters to be set depending on the options and methods.

2) When the screen is full, the scrolling is interrupted and can be restarted by typing <Return> or interrupted by q+<Return>. Inunterrupted scrolling is made available by SET/ECH SESS-DISP=NO

**Examples:** `none`

## SUBTRACT/BACKGROUND *stdred/echelle* 03-MAR-1991 PB

**Purpose:** Compute and subtract background from input frame and update descriptor BACKGROUND. The background estimate step will be skipped when running high level commands (FLAT/ECH, FILT/ECH, CALIB/ECH, RESP/ECH, REDU/ECH).

**Syntax:** `SUBTRACT/BACKGROUND input bkg output [bkgmtd] [bkgvisu]`

| | |
|---|---|
| `input` | name of raw input frame |
| `bkg` | name of background frame |
| `output` | name of output frame (input - bkg) |
| `bkgmtd` | Background computation method (POLY,SPLINE,SMOOTH) Echelle keyword is BKGMTD and default is POLY. |
| `bkgvisu` | Optional visualization of the background (YES/NO) (plots central column of input and bkg) Echelle keyword is BKGVISU and default is YES. The required graphic window is created if not already existent. The limits of the plot must be previously set using SET/PLOT command. |

**Note:** The options and parameters of this high level command are read from section 2. (Background) of the SHOW/ECHELLE. The current method is defined by keyword BKGMTD (possible values POLY, SPLINE, SMOOTH) and parameters to be set for each method are displayed by the SHOW/ECHELLE.

See also BACKGROUND/ECHELLE, BACKGROUND/SMOOTH, SELECT/BACKGR

**Examples:** `none`

# TUTORIAL/ECHELLE

## TUTORIAL/ECHELLE *stdred/echelle* 25-APR-1991 PB

**Purpose:** Demonstrates main commands of echelle package

**Syntax:** `TUTORIAL/ECHELLE`

**Note:** a) This procedure requires a display and a graphic window. They are automatically created if necessary.

b) Frames casff.bdf, casth.bdf, casobj.bdf, casstd.bdf are copied from MID_TEST if not already present. Tables lincat06.tbl (renamed thar.tbl) and ltt1020.tbl are copied from MID_CASPEC if not already present.

c) The only interactive part of the tutorial consists of identifying lines for the wavelength calibration. The user will be prompted to identify the occurences of two lines in adjacent orders:
Orders 88/89 Wavelength: 6416.315 Orders 98/99 Wavelength: 5760.550
Order numbers are displayed on the left side and wavelengths on the right side of the display window.

The execution of the tutorial is stopped, in order to enable a command like:
COPY/DISPLAY ps_printer

The tutorial is reactivated by the command:
CONTINUE

d) The four main steps of echelle package are performed:
CALIBRATE/ECHELLE FLAT/ECHELLE RESPONSE/ECHELLE REDUCE/ECHELLE

During the wavelength calibration, if the sequence of identification is correct, an angle of rotation is computed and must be around -0.09 degrees. Then the user must provide as input an order number (88) and two wavelengths (6416.315 and 5760.550).

It is recommended to consult Vol. B, Chapter Echelle Package, before running the tutorial.

**Examples:** `none`

## UPDATE/ECHELLE *stdred/echelle* 09-MAR-95 PB

**Purpose:** This command is called by VERIFY/ECHELLE (which is itself a low-level echelle command) to update the session keywords describing the geometry of the echelle spectrum.

**Syntax:** `UPDATE/ECHELLE image`

`image` Image name.

**See also:** VERIFY/ECHELLE, UPDATE/ORDER

**Note:** None.

**Examples:** `UPDATE/ECHELLE ccd0026`

## UPDATE/ORDER *stdred/echelle* 09-MAR-95 PB

**Purpose:** This command updates the order definition coefficients and the background table by offseting the positions. The offset can be provided numerically or be determined automatically.

**Syntax:** `UPDATE/ORDER image [offset]`

| | |
|---|---|
| `image` | Image name. |
| `offset` | Numerical value of the offset or AUTO (default).<br>In mode AUTO, the command OFFSET/ECHELLE is invoked to determine the offset value. |

**See also:** OFFSET/ECHELLE, UPDATE/ECHELLE, VERIFY/ECHELLE.

**Note:** None.

**Examples:** `UPDATE/ORDER ccd0026 12.`

## VERIFY/ECHELLE *stdred/echelle* 30-JUL-1991 PB

**Purpose:** Low level command of echelle package. Called by many commands to check input frames.

**Syntax:** `VERIFY/ECHELLE file [type]`

| | |
|---|---|
| `file` | input file |
| `type` | Can be FRAME (default), TABLE, EXTR, REBIN |

**Note:** The size of the images are checked against the value of the following echelle keywords:

IMSIZE/I/1/2 Size of raw frames (number of col., rows)
ECHORD(1) Number of orders
SCAN/I/1/2 Limit of scanned part of raw frames

**Examples:** `none`

# Appendix E

# Standard Reduction - ccdred

# Standard Reduction Commands

# Context: ccdred

# ALIGN/MOSAIC

## ALIGN/MOSAIC *stdred/ccdred* 19-Jul-1995 RHW

**Purpose:** Align the elements of the mosaiced frame

**Syntax:** `ALIGN/MOSAIC in_frm in_tab out_frm method,data [xref,yref] [xoff,yoff] [x_size,y_size]`

`in_frm` The mosaiced frame to be aligned. This frame must have been produced by the CREATE/MOSAIC command and have an accompanying database table specified by parameter in_tab.

`in_tab` The database table from the CREATE/MOSAIC task.

`out_frm` The aligned output frame produced by the command.

`method,data` Method to compute the matching intensities in the overlap regions. Three inputs are possible: a) S(hift),xshift,yshift. In this case the shift is assumed to be constant with respect to its neighbour in x (xshift) and in y (yshift). b) C(oord),table. In this case, the table is assumed to contain the coordinates of objects in the input frame, one object per line in the following format:
1) the x and y coordinates of the object in the first subraster;
2) the x and y coordinates of the same object in the second subraster; 3) the x and y coordinates of the next object in the first subraster; etc.
c) (R)eference,table. The table is assumed to contain the x and y shifts in columns 1 and 2 respectively for each input subraster relative to the reference subraster. The most common use of this option is to make fine adjustments by hand to the output of ALIGN/MOSAIC by editing the computed shifts slightly and rerunning the command with the new shifts.

`nxrsub,nyrsub` The column and row index of the reference subraster. This will default to the central subraster.

`xref,yref` The x and y offset of the destination in the output frame of the reference subraster. By default the reference subraster occupies the same position in the output frame that it does in the input frame.

`nocols,norows` The number of columns and lines in the output frame. The defaults are the number of columns and lines in the input frame.

**Note:** The command CREATE/MOSAIC is controlled by the input parameters and an additional number of CCD keywords:
MO_TRIM - the number of columns or rows to trim off each edge of each input subraster before inserting it in the output frame. The default is to trim one column or row at each edge.
MO_INTER - The type of interpolant use to shift the subraster. Possible options area : nearest, linear, poly3, poly5, and spline3.
MO_BLANK - to define the pixel intensity in undefined regions.

ALIGN/MOSAIC takes the mosaiced frame (in_frm), the database file (in_tab) generated by CREATE/MOSAIC, and a list of coordinates (method) and computes an output frame (out_frm) in which all the individual subrasters are aligned. If method="coords", ALIGN/MOSAIC accumulates the relative shifts between adjacent subrasters from the table into a total shift for each subraster with respect to the reference subraster.

Shifts which do not correspond to adjacent subrasters are ignored. For subrasters which have no direct shift information, ALIGN/MOSAIC makes a best guess at the x and y shift based on the shifts of nearby subrasters which do have direct shift information. If the x and y shifts are sufficiently uniform over the whole input frame the user may set the method parameter to "shifts" and supply values for the xshift and yshift. Alternatively ,the shifts may be read from a table file using the method="F,Table".

Coordinate tables may be generated interactively using the command SHIFT/MOSAIC command and using the display cursor.

The subrasters are inserted into the output frame using the interpolation scheme defined by the CCD keyword MO_INTER, and aligned with reference to the subraster defined by the parameter nxrsub and nyrsub, using the shifts defined by the coordinates in a table or defined by xshift and yshift parameters. Subrasters are inserted into the output frame in the order they were placed in the original mosaic with pixels in the most recently placed subrasters replacing those in earlier placed ones in the overlap regions. Undefined pixels in the output frame are given the value stored in the keyword MO_NUL.

The position of the reference subraster in the output frame may be adjusted by setting the offset parameters xref and yref. The edges of each subraster may be trimmed before insertion into the output frame by setting the CCD keyword MO_TRIM.

**See also:** SET/CCD, SHOW/CCD, CREATE/MOSAIC, SHIFT/MOSAIC, MATCH/MOSAIC, FIT/MOSAIC

**Examples:**
```
ALIGN/MOSAIC mosin mosout mosdb coords,coordtab 6,5
```
Align the mosin frame with respect to subraster 6, 5

```
SET/CCD MO_TRIM=2,2,2,2
```
ALIGN/MOSAIC mosin mosout mosdb coords,coordtab 6,5 The same as above but trim 2 rows and columns off of each input image before inserting into the output frame.

# BIAS/CCD

## BIAS/CCD *stdred/ccdred* 29-Mar-1993 RHW

**Purpose:** Correct the input frame for the bias offset using a bias frame

**Syntax:** `BIAS/CCD [in_fram] [out_fram] [bs_fram]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; No default. |
| `out_fram` | output frame resulting from subtracting the bias frame from the input frame; no default. |
| `bs_fram` | bias frame to be subtracted from input frame; default is the name stored in the keyword SC_BSFRM. |

**Note:** This command provides a simple subtraction of the bias offset from a CCD frame. More advanced methods include fitting of the bias offset averaged over a number of rows or columns, and subtracting this fit result from all columns/rows of the science frame.

In case the bias frame does not exit, a fatal error will be issued. In such case you may generate the bias frame using COMBINE/CCD.

The command exits if the input frame has already been bias corrected. The descriptor CCDSTAT in the output frame is updated to indicated that the bias offset has been subtracted.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, OVERSCAN/CCD, COMBINE/CCD

**Examples:** `BIAS/CCD susi0097 susi97bscor susibias`
Subtract the bias frame 'susibias' from the input frame susi0097. The output frame is susi97bscor.

COMBINE/CCD *stdred/ccdred* 1-Nov-1993 RHW

**Purpose:** Combined a number of CCD frames of the same exposure type

**Syntax:** `COMBINE/CCD exp [in_spec] [out_fram]`

| | |
|---|---|
| `exp` | exposure code of the frames to be combined. Options area:<br>BS for BIAS frames;<br>DK for DARK frames;<br>FF for FLAT field frames;<br>SK for SKY frames;<br>OT for OTHER frames.<br>The parameter 'exp' determines which keyword setting will be used for the combining (see below). |
| `in_spec` | a) name of a MIDAS table (with the extension '.tbl'), or// b) name of a MIDAS catalogue, or c) string of input single calibration frame, separated by comma's. By default the input is taken from the keyword CCD_IN. |
| `out_fram` | resulting combined output frame. No default output. |

# COMBINE/CCD

**Note:** In case a MIDAS table is used the command expects the name of the output master calibration frames in the column which name should be available from the keyword 'exp'_COL. The name of this output master calibration frame should conform to the Data Organizer/CCD naming convention. Obviously, the individual calibration frames must available.

The parameters needed for the combining are read from the CCD keyword structure. This keyword setting can be consulted by the command SHOW/CCD 'exp_typ', where 'exp' can be BS, DK, FF, or SK. The listing shows these keywords, their options and meaning:
CCD_IN input spec - input table or catalogue
'exp'_SIG yes/no - create sigma and pixel count image?
'exp'_TYP all/exp. type - exposure time descriptors allowed
'exp'_MET method - method of combining (see below)
'exp'_RAN value1,value2 - range of pixel values allowed
'exp'_DEL yes/no - delete input frames afterwards?
'exp'_EXP yes/no - scale by exposure time?
'exp'_STA mean/median/mode - statistics for scaling or offset
'exp'_SCA yes/no - scaling required?
'exp'_OFF yes/no - offset correction required?
'exp'_WEI yes/no - use a weighted average?
'exp'_SEC area - area for computing the statistics
'exp'_CLP value1,value2 - low/high sigma clipping factor
'exp'_BLA value - blank value for rejections

The combining will be done according to the method contained in the keyword 'exp'_MET. Currently, the following methods are supported:
sum - simple sum of all input input images;
ave(rage) - input images are combined by averaging;
median - input images are combined by medianing each pixel;
min(reject) - reject the minimum value from the average;
max(reject) - reject the maximum value from the average;
minmax(reject) - reject the minimum and maximum from the average;
sig(clip) - apply sigma clipping algorythm to each pixel;
avsig(clip) - apply sigma clipping algorythm together with minmax rejection;
For more details about the various methods, please refer to Chapter Direct Imaging in Volume B of the ESO-MIDAS Users' Guide.

Before the combining is done the exposure type descriptors of the input frame are compared with the descriptor type stored in the keyword 'exp'_TYP. In case this keyword is filled with '*' or '?' all exposure types are allowed. Else, a fatal error will follow if the the keyword content is not equal to one or more frame exposure types.

The first and second element of descriptor LHCUTS, containing the real minimum and maximum of the frame are updated. In addition, the descriptor MEAN, MODE and MEDIAN contain the corresponding values of the pixels values within the area defined by the keyword IM_SEC, the useful image section.

To generate the output combined frame only those input pixels will be used that have intensities within the range stored in the keyword 'exp'_RAN. The sigma frame, if created, also has the rejected pixels excluded.

In case the keyword 'exp'_SIG is set two additional frame will be created: one containing the number of frames used for the combining result; a second contains the standard deviation around the mean result.

**See also:** SET/CCD, SHOW/CCD, AVERAGE/IMAGE, AVERAGE/WINDOW

**Examples:** `COMBINE/CCD ff ost_asso.tbl`
Create all master flat frames which names are stored the master flat column in the table ost_assoc.tbl.

`COMBINE/CCD bs myflats.cat`
Created the combined flat from all the frames in the catalogue myflats.

`COMBINE/CCD bs myflat1,myflat1,myflat2`
Created the combined flat from the three input frames.

`COMBINE/CCD bs`
Combine bias frames. The input in taken from the keyword CCD_IN. Depending on the input continue like in one the above example.

# CREATE/MOSAIC

## CREATE/MOSAIC *stdred/ccdred* 19-Jul-1995 RHW

**Purpose:** Mosaic a set of (infrared) ccd frames

**Syntax:** `CREATE/MOSAIC in_cat out_frm out_tab nx_sub,ny_sub`

`[not1,not2,...] [nocol,norow]`

`in_cat` The input catalogue of input subframes to be mosaiced. The frames are assumed to be ordered either by row, column, or in a raster pattern. If the frame catalogue is not in order the user must correct it or create a new catalogue. The frames in the input catalogue are assumed to have all the same size. No default.

`out_frm` The name of the output frame. No default.

`out_tab` The name of the MIDAS table that will contain subframes and parameters produced by the command. This table can be used as input for the command ALIGN/MOSAIC and MATCH/MOSAIC.

`nx_sub,ny_syb` The number of subrasters along a row of the output frame (nx_sub), and the number of subrasters along a column of the output frame (ny_sub). No default.

`[not1,not2,...]` The list of unobserved subrasters. For example, if the subrasters 3 to 5 and 10 of a sequence of observations were not observed then the input should be 3:5,10. The number of unobserved subrasters plus the number of frames in the input catalogue must equal to nx_sub*ny_sub. No default.

`[nocol,norow]` The number of columns and rows in the output frame. If nocols is 0 then the program will compute the number of columns using the size of the input subrasters. Default 0,0.

**Note:** The command CREATE/MOSAIC is controlled by the input parameters and an additional number of CCD keywords:
MO_SEC - area in the input subraster to determine the mean;
MO_SUBT - to switch on/off ther subtraction of the mean;
MO_CORN - to define the starting corner (ll, lr, ul, ur);
MO_DIREC - to define the direction of writing (row or column);
MO_RAST - to add subrasters in a raster pattern, or to returning to the start of the column or row;
MO_OVER - to define the number of blank lines between subrasters;
MO_TRIM - to trim columns and rows of the the subrasters;
MO_BLANK - to define the pixel intensity in undefined regions.
CREATE/MOSAIC takes a catalogue of subrasters with identical dimensions and combines them into a single output frame. The order in which the subrasters are placed in the output frame is determined by the CCD keywords MO_CORN, MO_DIREC, and MO_RAST. The size of the individual subrasters in the output frame may be altered by setting the MO_TRIM keyword.
CREATE/MOSAIC uses the subraster size, the number of subrasters, the MO_NOVER keywords, and the nx_sub and ny_sub parameters to compute the size of the output frame. A frame of size larger than the minimum required can be specified by setting nocols and norows parameters. Undefined regions of the output frame are given the value MO_NUL.
The median of a section each subraster may be optionally computed and placed in the database file by setting MO_SEC keyword. The computed median will be subtracted from the input subrasters if the keyword MO_SUBT is set to yes.

**See also:** SET/CCD, SHOW/CCD, SHIFT/MOSAIC, OFFSET/MOSAIC, ALIGN/MOSAIC, MATCH/MOSAIC

**Examples:** `CREATE/MOSAIC in_cat mosaic mosaic.dat 8,8`

Mosaic a catalogue of 64 infrared frame onto an 8 by 8 grid in column order starting in lower list corner. Allow one blank column and row between each subraster.

`CREATE/MOSAIC in_cat mosaic mosaic.dat 8,8 3,9`
Mosaic a list of 62 infrared frame onto an 8 by 8 grid in column order starting in the lower left corner. Allow one blank column and row between each subraster. Subrasters 3 and 9 in the sequence do not exist and are to be replaced in the output frame with the unknown value stored in the keyword MO_NUL.

---

DARK/CCD *stdred/ccdred* 29-Mar-1993 RHW

**Purpose:** Correct input frame for dark current offset using a dark current frame

**Syntax:** `DARK/CCD [in_fram] [out_fram] [dk_fram]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; default is the name stored in the keyword CCD_IN. |
| `out_fram` | output frame resulting from the weighted subtracting of the dark frame from the input frame; no default |
| `dk_fram` | dark frame to be subtracted from input frame; default is the name stored in the keyword SC_DKFRM. |

**Note:** The command subtracts the scaled dark current frame from the input frame. The scaling factor is O_TIME1/O_TIME2, where O_TIME1 is the exposure time of the input frame, and O_TIME2 is the exposure time of the dark current frame.

In case the dark frame does not exit, a fatal error will be issued. In such case you may generate the bias frame using COMBINE/CCD.

Both, the input and dark current frame should contain the descriptors for the exposure time (see keyword 'O_DESCR' for the actual name of the descriptor. If they are non-existing or contain invalid values (less or equal to zero) an error message is given and an exist follows.

The command will exit if the dark correction has already been done. After the command has finished the descriptor CCDSTAT in the output frame is updated to indicated that the dark has been subtracted.

**See also:** SET/CCD, SHOW/CCD, COMBINE/CCD, REDUCE/CCD, COMPUTE/IMAGE

**Examples:** `DARK/CCD susi0097 susi97dkcor susidark`
Subtract the dark frame 'susidark' from the input frame susibs0097. The output frame is susi97dkcor.

# FIT/MOSAIC

## FIT/MOSAIC *stdred/ccdred* 20-Sep-1995 RHW

**Purpose:** Align and match the elements of the mosaiced frame

**Syntax:** `FIT/MOSAIC in_frm in_msk in_tab out_frm [match] [nxrsub,nyrsub] [xref,yref] [x_size,y_size]`

`in_frm` The mosaiced frame to be aligned. This frame must have been produced by the CREATE/MOSAIC command.

`in_msk` A input image containing information which of the pixels of the input subrasters are considered bad and hence should be excluded for the matching process. Valid data points should have the values 1. The size of the image must the identical to the size of the individual input subrasters. Default all pixels should be used.

`in_tab` The database table from the CREATE/MOSAIC task. This table contains all relevant information about the subrasters and the way the mosaic is built.

`out_frm` The aligned output frame produced by the command. No default.

`match` Match intensities using the overlap region between adjacent subrasters. The median intensity is computed in the overlap region and the intensity scale of the current subraster is scaled to that of the previous subraster. Intensities are matched in two dimensions, first in the order in which they were placed in the output image and then in the orthogonal dimension. The default is "*", i.e. match everything. Those subrasters to be matched must be listed by number. For example to match intensities for subrasters 1 to 5, 10 and 20 set match = "1:5,10,20". To match all the subrasters set match = "*".

`nxrsub,nyrsub` The column and row index of the reference subraster. This will default to the central subraster.

`xref,yref` The x and y offset of the destination in the output frame of the reference subraster. By default the reference subraster occupies the same position in the output frame that it does in the input frame.

`nocols,norows` The number of columns and lines in the output frame. The defaults are the number of columns and lines in the input frame.

**Note:** Like the ALIGN and MATCH commands, FIT/MOSAIC is controlled by the input parameters and an additional number of CCD keywords:
MO_TRIM - the number of columns or rows to trim off each edge of each input subraster before inserting it in the output frame. The default is to trim 1 column or row at each edge.
MO_INTER - The type of interpolant use to shift the subraster. Possible options area : nearest, linear, poly3, poly5, and spline3.
MO_BLANK - to define the pixel intensity in undefined regions.
FIT/MOSAIC is able to a mosaicing process taking into account all information that is available to determine the best estimate for the offsets between the individual frames. Rather then having a bootstrap approach like in the command MATCH/MOSAIC the FIT/MOSAIC command uses all offsets between all frames.
The commands does the mosaicing in a two step approach. The first one is to determine the offset between the single subraster in their overlapping regions. In the second step this information is then used to do a least square fit.
In the first step the overlapping areas are obtain from the input database table. Within the overlapping regions the median intensities are determined and the offsets of these median values calculated. Bad pixels indicated by the bad pixel input image will not be taken into account. Also, a pixel will only be used in the calculation of the median if its corresponding pixel is also used in the other frame.
In the second step the relative offsets are used to determine the best fitting background offset for each frame. Using the "match" input parameter the user can excluded one or more subrasters from the fit. In addition to the output file, the command will output the offset for each subraster and the square root of the sum of the residuals and the standard deviation of the residuals. From this information bad subraster can be identified and excluded from the fit. The best result may been obtained by continuing this process until all frames more than 3 sigma above the mean have been excluded.

**See also:** SET/CCD, SHOW/CCD, CREATE/MOSAIC, SHIFT/MOSAIC, ALIGN/MOSAIC, MATCH/MOSAI(

**Examples:** `FIT/MOSAIC mosin ? mosdb mosout`
Align and match the mosin frame with respect to default reference subraster. Use the table mosdb for getting the overlap regions. The output frame is mosout. All pixels will be taken into account.

`FIT/MOSAIC mosin mosmsk mosdb mosout 3,4 6,5`
The same as above but the reference frame in the mosaic is now (subx=6, suby=5). The vaid pixels will bne taken from the mask mosmsk. the subrasters 3 and 4 will not be taken into account in the fitting.

# FIXPIX/CCD

| FIXPIX/CCD | *stdred/ccdred* | 31-Mar-1993 RHW |
|---|---|---|

**Purpose:** Do a correction of bad pixels in the input frame

**Syntax:** `FIXPIX/CCD [in_fram] [out_fram] [fix_table] [fix_meth]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; default is the name stored in the keyword CCD_IN. |
| `out_fram` | output frame after correction has been applied; no default. the name stored in the keyword CCD_OUT. |
| `fix_table` | MIDAS table containing the bad pixels. The format of the table is dependent of which correction option is used. It should contain the following columns with the coordinate information (in world coordinate units):<br>for option 'A' columns :XSTART, :YSTART, :XEND and :YEND;<br>for option 'P' columns :XSTART, :YSTART, :XEND and :YEND;<br>for option 'C' column :X<br>for option 'R' column :Y<br>The default table name is the one stored in the CCD keyword FX_TABLE. |
| `fix_meth` | method to be applied for fixing the bad pixels. The option is derectly related withg the MIDAS coomand that will be used for the correction. Possible choises are:<br>'AREA': for applying the command MODIFY/AREA;<br>'PIXEL': for applying the command MODIFY/PIXEL.<br>'COLUMN': for applying the command MODIFY/COLUMN;<br>'ROW': for applying the command MODIFY/ROW;<br>The default option is the one stored in the CCD keyword FX_METH. |

**Note:** A number of additional parameters values are needed if order to make the correction succesful. These additional parameters are obtained from the CCD keyword structure; the number and their meaning depend on the option (method) that will be used. Below follows the possible options, the corresponding MODIFY commands used, the parameters needed and the keywords where the parameters are expected to be stored (using SET/CCD)
method 'A' (MODIFY/AREA): parameter 'degree'; FX_FPAR(1)
parameter 'constant'=0; no keyword;
method 'P' (MODIFY/PIX): parameter 'arfacts'; FX_FACT
parameter 'xdeg,ydeg,niter'; FX_FPAR
parameter 'noise'; FX_NOISE.
method 'C' (MODIFY/COL): parameter 'col_type'=V; no keyword;
method 'R' (MODIFY/ROW): parameter 'row_type'=V; no keyword;

The command will not try to correct the input frame if that has already been done. After the command has finished the descriptor CCDSTAT in the output frame is updated to indicated that the bad pixel correction has been applied.

A bad pixel table can, if not provided with the instrument setting, can be created by the command GET/CURSOR, using the cursor option for one or two cursors.

**See also:** MODIFY/AREA, MODIFY/COLUMN, MODIFY/ROW, MODIFY/PIXELS, GET/CURSOR, SET/CCD, SHOW/CCD, LOAD/CCD, REDUCE/CCD

**Examples:** `FIXPIX/CCD amsterdam rotterdam P groningen`
Fix the bad pxiels in the ccd frame amsterdam using the command MODIFY/PIXELS and the input bad pixel table groningen. The output frame will be rotterdam.

FLAT/CCD *stdred/ccdred* 29-Mar-1993 RHW

**Purpose:** Do a flat field correction of the input frame

**Syntax:** `FLAT/CCD [in_fram] [out_fram] [ff_fram]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; default is the name stored in the keyword CCD_IN. |
| `out_fram` | output frame resulting from the devision of the input frame by a scaled flat field; no default. |
| `ff_fram` | flat frame to be used; default is the frame stored in the keyword SC_FFFRM. |

**Note:** The command devides the input frame by the flat field frame according to the formula:
out_fram = in_fram/(ff_fram/mean)
where mean is the value of the descriptor 'MEAN' in the flat field frame. This descriptor is created by the command COMBINE/CCD. Its value is computed over the useful image defined in the keyword 'IM_SEC'. In case the descriptor does not exist, the mean intensity over the flat field using the area in the key 'IM_SEC' will be used as the scaling factor. In case the descriptor contains a number smaller or equal to zero the value 1.0 will be used.

In case the flat frame does not exit, a fatal error will be issued. In such case you may generate the flat frame using COMBINE/CCD.

The command exists if the flat fielding has already been done. After the command has finished the descriptor CCDSTAT in the output frame is updated to indicated that the flat field correction offset has been applied.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, COMPUTE/IMAGE, COMBINE/CCD

**Examples:** `FLAT/CCD susi0097dkcor susi97ffcor susiflat`
Devide susu0097dkcor by the flat field 'susiflat'. The output frame is susi97ffcor.

# FRCOR/CCD

| FRCOR/CCD | *stdred/ccdred* | 20-Dec-1993 | RHW |
|---|---|---|---|

**Purpose:** Make fringe correction frame(s0 from sky frames

**Syntax:** `FRCOR/CCD [in_spec] [out_frm] [xboxmn,xboxmx] [yboxmn,yboxmx]`

`[clip] [lowsig,higsig]`

`in_spec` single frame or MIDAS CCD reduction table (with extension .tbl) containing the names of the sky frames from which the fringe correction frames should be created. In case the input is a reduction table the names of the input frames will be obtained from the column :SKY in that table (see below). Default input is taken from the keyword CCD_IN.

`out_frm` name of the output fringe correction frame. Default output is taken from the keyword CCD_OUT. This parameter is only used is case of single frame input for the first parameter. In case the input is a ccd reduction table the name(s) of the output frame(s) will be read from tke column :FR in this table (see below).

`xboxmn,xboxmx` minimum and maximum smoothing box size along the x axes. The minimum box size is used at the edges and grows to the maximum size in the middle of the image. This allows the smoothed image to better represent gradients at the edge of the image. If a size is less then 1 it is interpreted as a fraction of the image size. If a size is greater than or equal to 1 then it is the box size in pixels. A size greater than the size of image selects a box equal to the size of the image. Default values are taken from the keyword IL_XBOX.

`yboxmn,yboxmx` see above. Default values are taken from IL_YBOX.

`clip` Clean the input frame(s) of objects? If yes then a clipping algorithm is used to detect and exclude objects from the smoothing. Default taken from IL_CLIP.

`lowsig,higgsig`ma clipping thresholds above and below the smoothed sky illumination frame. Default values are taken from the keyword IL_SIGMA. The keyword is only read in case clip=yes.

**Note:** The input sky frames are automatically processed up through flat fielding and illumination before computing the fringe pattern.

In case the input is the CCD reduction table the command will create fringe correction frames for all selected data sets (flat frames) in the reduction table, provided the sky column (:SKY) contains existing sky frames. The name of the output illumination frames are obtained from the column :FR. Relevant calibration frame will be taken from the other calibration column belonging to the same data set.

The effects of objects in the sky frames may be minimized by using a sigma clipping algorithm to detect and exclude the objects from the average.

The command operates very similar to the command REDUCE/CCD, which the exception that it does not check for the exposure type of the input frame. Master frames to be used to calibrate the input frame first are taken for the SC_ keywords (in case of single frame input), or from the reduction table in case of table input. For details see the command REDUCE/CCD.

The calibrated sky frame is heavily smoothing to obtain the sky illumination frame using a moving "boxcar" average. This smoothed frame is subtracted from the unsmoothed frame to yield the output fringe frame. The fringe frame can used by REDUCE/CCD to remove the fringe pattern from the science frames.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD, REDUCE/CCD, ILLSKY/CCD

**Examples:** `FRCOR/CCD redtbl.tbl`
Reduce all flat frames in the flay field column in the ccd reduction table redtbl.tbl. Master calibration frames will be obtained from that reduction table. Do the calibration according to the keyword settings. After the calibrations are done create the fringe frames, using the 'IL_' and 'FR_' keyword settings. Calibration frames will be taken from the table redtbl.tbl

`FRCOR/CCD m100rraw m100rraw_fr`
Start the ccd reduction sequence for the frame m100rraw. This input frame should be present. The output frame will be m100rraw_fr. In case only dark correction is to be applied the names of this calibration frame will be obtained from the keyword SC_DKFRM.

# FRINGE/CCD

## FRINGE/CCD *stdred/ccdred* 27-August-1993

**Purpose:** Do a fringe correction of the input frame

**Syntax:** `FRINGE/CCD [in_fram] [out_fram] [fr_fram]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; default is the frame stored in the keyword CCD_IN. |
| `out_fram` | output frame resulting from the devision of the input frame by a scaled illumination field; no default. |
| `fr_fram` | fringe frame by which input frame is to be divided; default is the frasme storedf in the keywords SC_FRFRM. |
| `fr_scale` | fringe frame scaling factor. Default is the number stored in descriptor FRINGESCALE of the fringe frame. See below. |

**Note:** The command devides the input frame by the flat field frame according to the formula:
out_fram = in_fram - fr_scale*obstimi/obstimf*fr_fram
where fr_scale is the fringe scaling factor, and ostimi and obstimf are the exposure time of the input and fringe frame respectively. By default fringe scale factor is read from the fringe frame descriptor FRINGESCALE. If no value can be obtained the default will be 1.0. In case the descriptor contains a number smaller or equal to zero the value 1.0 will be used.

In case the fringe frame does not exit or the frame cannot be used as an fringe frame a fatal error will be issued.

The command exists if the fringe correction was already applied. After the command has finished the descriptor CCDSTAT in the output frame is updated to indicated that the frame is fringe corrected.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, MKFRCOR/CCD

**Examples:** `FRINGE/CCD susi0097 susi97frcor susi0097_fr`
Correct the frame susi0097 for fringing. The frame susi0097_fr is to be used as fringe frame; output is the frame susi0097frcor. The fringe scaling factor will be obtained from the descriptor of the fringe frame.

HELP/CCD *stdred/ccdred* 1-Nov-1993 RHW

**Purpose:** Show the parameter setting of the current CCD session

**Syntax:** `HELP/CCD [key_string]`

`key_string` usually, the complete name of a CCD keyword. However also parts of keyword names can be given as input.
If no parameter is entered, a general help message is displayed.

**Note:** Both the current value(s) of the keyword and its original default content are displayed.

**See also:** SHOW/CCD, SET/CCD, READ/KEY

**Examples:** `HELP/CCD OV_REJEC`
Display the contents of the keyword OV_REJEC

`HELP/CCD BS`
Display information about all keywords that contain the string BS.

`HELP/CCD`
Display the general command information of the CCD package

# ILLCOR/CCD



**Purpose:** Make flat field illumination correction frame(s)

**Syntax:** `ILLCOR/CCD [in_spec] [out_frm] [xboxmn,xboxmx] [yboxmn,yboxmx]`

`[clip] [lowsig,higsig]`

`in_spec` single flat frame or MIDAS CCD association table (with the extension .tbl) containing the names of the flat frame(s) in the column :FF. Default input is taken from the keyword CCD_IN.

`out_frm` name of the output illumination frame. Default output is the name of the input table with extension "_ill". Only used in case of single frame input for the first parameter. In case the input is a MIDAS table the name(s) of the output frame(s) will have the same name as the sky frame field expanded with '_ill' (see below).

`xboxmn,xboxmx` minimum and maximum smoothing box size along the x axes. The minimum box size is used at the edges and grows to the maximum size in the middle of the image. This allows the smoothed image to better represent gradients at the edge of the image. If a size is less then 1 it is interpreted as a fraction of the image size. If a size is greater than or equal to 1 then it is the box size in pixels. A size greater than the size of image selects a box equal to the size of the image. Default values are taken from the keyword IL_XBOX.

`yboxmn,yboxmx` see above. Default values are taken from IL_YBOX.

`clip` Clean the input frame(s) of objects? If yes then a clipping algorithm is used to detect and exclude objects from the smoothing. Default taken from IL_CLIP.

`lowsig,higgsig` sigma clipping thresholds above and below the smoothed illumination. Default values are taken from the keyword IL_SIGMA. The keyword is only read in case clip=yes.

**Note:** The command operates very similar to the command REDUCE/CCD, which the exception that the command not check for the exposure type of the input frame.

In case the input is an association table the command will check for the existence of the flat frames. Flat frames that do not exit will be created first, provided the association table contain the flat column. Other relevant calibration frames for processing the flat frames, will be taken from the corresponding calibration columns belonging to the same science frame. Output illumination frame(s) will have the same name as the original flat frame except for the extension '_ill'.

In the case of single frame input and all calibration frames input will be taken from the SC_ keywords and must exist.

The input flats are automatically processed up through dark current calibration before computing the illumination pattern. The illumination correction, the inverse of the illumination pattern, is applied by the CCD package to remove the illumination pattern introduced by the flat field. The combination of the flat field calibration and the illumination correction based on the flat field is equivalent to removing the illumination from the flat field (see ILLFLAT/CCD). This two step calibration is generally used when the observations have been previously flat field calibrated. This task is closely related to SKYCOR/CCD which determines the illumination correction from a blank sky frame; this is preferable to using the illumination from the flat field as it corrects for the residual illumination error.

The illumination frame is produced by heavily smoothing the calibrated flats frames using a moving "boxcar" average. The effects of objects in the frames may be minimized by using a sigma clipping algorithm to detect and exclude the objects from the average. The output illumination frame can used by REDUCE/CCD to remove the illumination pattern in the science frames.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD, REDUCE/CCD, SKYFLAT/CCD

**Examples:** `ILLCOR/CCD ccd_asstbl.tbl`
Reduce all flat frames in the flat field column in the ccd table ccd_asstbl.tbl. Master calibration frames will be obtained from the same table. Do the calibration according to the keyword settings. After the calibrations are done create the illumination frames, using the 'IL_' keyword settings. Calibration frames will be taken from the table redtbl.tbl

`ILLCOR/CCD m100rraw m100rraw_il`
Start the ccd reduction sequence for the frame m100rraw. This input frame should be present. The output frame will be m100rraw_il. In case only dark correction is to be applied the name of this calibration frame will be obtained from the keyword SC_DKFRM.

# ILLFLAT/CCD

## ILLFLAT/CCD *stdred/ccdred* 1-Nov-1993 RHW

**Purpose:** Apply correction to flat field to remove illumination pattern

**Syntax:** `ILLFLAT/CCD [in_spec] [out_frm] [xboxmn,xboxmx] [yboxmn,yboxmx]`

`[clip] [lowsig,higsig]`

`in_spec` single flat frame or MIDAS CCD reduction table (with the extension .tbl) containing the names of the flat frame(s) in the column :FF. Default input is taken from the keyword CCD_IN.

`out_frm` name of the illumination corrected flat field. Default the output file the same as the input frame: hence by default the input flat frame will be OVERWRITTEN. This parameter is only used is case of single frame input for the first parameter. In case the output is taken a reduction table the output frame(s) will replace the input flat frames: hence the input flat frame will be OVERWRITTEN.

`xboxmn,xboxmx` minimum and maximum smoothing box size along the x axes. The minimum box size is used at the edges and grows to the maximum size in the middle of the image. This allows the smoothed image to better represent gradients at the edge of the image. If a size is less then 1 it is interpreted as a fraction of the image size. If a size is greater than or equal to 1 then it is the box size in pixels. A size greater than the size of image selects a box equal to the size of the image. Default values are taken from the keyword IL_XBOX.

`yboxmn,yboxmx` see above. Default values are taken from IL_YBOX.

`clip` Clean the input frame(s) of objects? If yes then a clipping algorithm is used to detect and exclude objects from the smoothing. Defaault taken from IL_CLIP.

`lowsig,higsig` sigma clipping thresholds above and below the smoothed illumination. Default values are taken from the keyword IL_SIGMA. The keyword is only read in case clip=yes.

**Note:** The command operates very similar to the command REDUCE/CCD, which the exception that the command not check for the exposure type of the input frame.

In case the input is an association table the command will check for the existence of the flat frames in the column :FF in the association table. Flat frames that do not exit will be created first, provided the association table contain the :FF column. Other relevant calibration frames for processing the flat frames, will be taken from the corresponding calibration columns. From the reduced flat field, an illumination correction frame is produced by smoothing the calibrated flat frame using a moving "boxcar" average.

The reduced flat frame is then divided by the scaled illumination correction to produce the illumination corrected flat. The output will OVERWRITE the original (reduced) flat frames. Hence, the output illumination flat will automatically be used as flat in subsequent calibration procedure(s). NOTE: THE MASTER FLAT FIELDS WILL BE SUBSTITUTED BY THE CORRECTED ONES.
In the case of single frame input and all calibration frames input will be taken from the SC_ keywords and must exist. Also here, the default is that the output illumination corrected flat will replace the input flat.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD, REDUCE/CCD, SKYCOR/CCD

**Examples:** `ILLFLAT/CCD redtbl.tbl`
Reduce all flat frames in the the ccd reduction table redtbl.tbl. All master calibration frames will be obtained from that reduction table. Do the calibration according to the keyword settings. After calibration produce an illumination frame, using the 'IL_' keyword settings. Hereafter the illumination corrected flat field is computed by dividing the original flat by the illumination frame. All calibration frames are taken from the table redtbl.tbl. The output corrected flats will have the same name as the original flat except for the extension '_if'.

`ILLFLAT/CCD m100rraw_ff m100rraw_if`
Do the illumination correction for the frame m100rraw_ff. Check if the input was already reduced, and do the reduction if needed. Hereafter produce the output output frame m100rraw_if. In case calibration frames are to be applied the names of these frames will be obtained from the keyword SC_xxFRM and SC_xxFRM, where 'xx stands for the exposure type of the calibration frame (BS, DK).

---

## ILLUMINATION/CCD

*stdred/ccdred* 27-August-1993

**Purpose:** Do an illumination correction of the input frame

**Syntax:** `ILLUMINATION/CCD [in_fram] [out_fram] [il_fram]`

| | |
|---|---|
| `in_fram` | input frame to be corrected; default is the frame stored in the keyword CCD_IN. |
| `out_fram` | output frame resulting from the devision of the input frame by a scaled illumination field; no default. |
| `il_fram` | illumination frame by which input frame is to be divided; default is the frame stored in the keyword SC_ILFRM. |

**Note:** The command devides the input frame by the flat field frame according to the formula:
out_fram = in_fram*mean/il_fram
where mean is the value of the descriptor 'CCDMEAN' in the illumination frame, created when the illumination frame produced. Its value is computed over the useful image defined in the keyword 'IM_SEC'. In case the descriptor does not exist, the mean intensity over the ilumination frame using the area in the key 'IM_SEC' will be used as the scaling factor. In case the descriptor contains a number smaller or equal to zero the value 1.0 will be used.

In case the illumination frame does not exit or the frame cannot be used as an illumination frame a fatal error will be issued.

The command exists if the illumination correction was already applied. After the command has finished the descriptor CCDSTAT in the output frame is updated to indicated that the frame is illumination corrected.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, ILLCOR/CCD, ILLFLAT/CCD, SKYCOR/CCD, SKYFLAT/CCD

**Examples:** `ILLUM/CCD susi0097dkcor susi97ilcor susi0097_il`
Devide susu0097dkcor by the scale illumination correction frame susi0097_il. The output frame is susi97ilcor.

# INIT/CCD

| INIT/CCD | *stdred/ccdred* | 1-Nov-1993 RHW |
|---|---|---|

**Purpose:** Initialize the CCD package, optionally using the setting of a saved session

**Syntax:** `INIT/CCD [name]`

`name` session name, previously saved

**Note:** INIT/CCD (without name) will also load the names of some standard ESO frame descriptors that contain instrument specific information about the observation into the CCD keyword structure. So, for example, the descriptors names for exposure type, exposure time, analog-to-digital value, read-out-noise are loaded into the CCD keywords.

By default, after initializing, the CCD context assumes that the standard ESO descriptors, originating from the ESO hierarchical FITS keywords are used.

These desciptor names are loaded by running a procedure eso_descr.prg. At initialization of the CCD context a copy of this procedure is put in the user's directory. If, for some reason, your descriptors names do not conform to the default ESO names, you can modify this small procedure, and execute is by "@@ eso_descr".

**See also:** HELP/CCD, SHOW/CCD, SET/CCD, LOAD/CCD, SAVE/CCD

**Examples:** `INIT/CCD`
Initialize the CCD package. This is automatically done if you enable the CCD context with SET/CONTEXT CCD

`INIT/CCD mysession`
Restore the complete settings of a previous CCD session, saved with the name 'mysession'.

LOAD/CCD *stdred/ccdred* 1-Nov-1993 RHW

**Purpose:** Load intrument/detector specifications into the CCD context

**Syntax:** `LOAD/CCD [instr]`

`instr` Name of the instrument/detector combination for which the default parameters will be loaded into the CCD context. The settings are read from the table eso_specs.tbl that, in case it does not exist, will be created in your directory.

**Note:** At start-up the CCD context creates keywords that will contain instrument and detector specifications parameters. However, at initialisation of the CCD context these keywords are not filled yet. The LOAD/CCD takes care of that by copying the default parameters for a specific telescope/instrument combination from a MIDAS table into the CCD keyword. These values will to be used as default values during the CCD reduction.

The parameters for ESO instrumentation are stored in the MIDAS table eso_specs.tbl. A copy of this table is stored in the user's working directory. If you want to change the numbers or to add more entries to this MIDAS table you can use the MIDAS table editor or other table commands.

**See also:** INIT/CCD, SET/CCD, SHOW/CCD, EDIT/TABLE, READ/TABLE, SHOW/TABLE

**Examples:** `LOAD/CCD eso_susi`
Load the instrumental specs for the ESO susi instrumental into the keyword structure.

`LOAD/CCD my_susi`
Load the (personal) instrumental specs for the susi instrumental into the keyword structure. The entry "my_susi" must be present in the eso_specs table in the user's working directory.

# MATCH/MOSAIC

## MATCH/MOSAIC *stdred/ccdred* 24-Jul-1995 RHW

**Purpose:** Align and match the elements of the mosaiced frame

**Syntax:** `MATCH/MOSAIC in_frm in_tab out_frm method,data [match] [xref,yref] [xoff,yoff] [x_size,y_size]`

`in_frm` The mosaiced frame to be aligned. This frame must have been produced by the CREATE/MOSAIC command.

`in_tab` The database table from the CREATE/MOSAIC task. This table contains all relevant information about the subrasters and the way the mosaic is built.

`out_frm` The aligned output frame produced by the command.

`method,data` Method to compute the matching intensities in the overlap regions. Three inputs are possible: a) S(hift),xshift,yshift. In this case the shift is assumed to be constant with respect to its neighbour in x (xshift) and in y (yshift). b) C(oord),table. In this case, the table is assumed to contain the coordinates of objects in the input frame, one object per line in the following format:
1) the x and y coordinates of the object in the first subraster;
2) the x and y coordinates of the same object in the second subraster; 3) the x and y coordinates of the next object in the first subraster; etc.
b) (R)eference,table. The table is assumed to contain the x and y shifts in columns 1 and 2 respectively for each input subraster relative to the reference subraster. The most common use of this option is to make fine adjustments by hand to the output of ALIGN/MOSAIC by editing the computed shifts slightly and rerunning the command with the new shifts.

`match` Match intensities using the overlap region between adjacent subrasters. The median intensity is computed in the overlap region and the intensity scale of the current subraster is scaled to that of the previous subraster. Intensities are matched in two dimensions, first in the order in which they were placed in the output image and then in the orthogonal dimension. The default is "*", i.e. match everything. Those subrasters to be matched must be listed by number. For example to match intensities for subrasters 1 to 5, 10 and 20 set match = "1:5,10,20". To match all the subrasters set match = "*".

`nxrsub,nyrsub` The column and row index of the reference subraster. This will default to the central subraster.

`xref,yref` The x and y offset of the destination in the output frame of the reference subraster. By default the reference subraster occupies the same position in the output frame that it does in the input frame.

`nocols,norows` The number of columns and lines in the output frame. The defaults are the number of columns and lines in the input frame.

**Note:** The command MOSAIC/MOSAIC is controlled by the input parameters and an additional number of CCD keywords:
MO_TRIM - the number of columns or rows to trim off each edge of each input subraster before inserting it in the output frame. The default is to trim 1 column or row at each edge.
MO_INTER - The type of interpolant use to shift the subraster. Possible options area : nearest, linear, poly3, poly5, and spline3.
MO_BLANK - to define the pixel intensity in undefined regions.

MATCH/MOSAIC takes the mosaiced frame (in_frm), the database file (in_tab) generated by CREATE/MOSAIC, and a list of coordinates (method) and computes an output frame (out_frm) in which all the individual subrasters are aligned. If method="coords", MATCH/MOSAIC accumulates the relative shifts between adjacent subrasters from the table into a total shift for each subraster with respect to the reference subraster.

Shifts which do not correspond to adjacent subrasters are ignored. For subrasters which have no direct shift information, MATCH/MOSAIC makes a best guess at the x and y shift based on the shifts of nearby subrasters which do have direct shift information. If the x and y shifts are sufficiently uniform over the whole input frame the user may set the method parameter to "shifts" and supply values for the xshift and yshift. Alternatively ,the shifts may be read from a table file using the method="file".

Coordinate tables may be generated interactively using the command SHIFT/MOSAIC command and using the display cursor.

The subrasters are inserted into the output frame using the interpolation scheme defined by the CCD keyword MO_INTER, and aligned with reference to the subraster defined by the parameter nxrsub and nyrsub, using the shifts defined by the coordinates in a table or defined by xshift and yshift parameters. Subrasters are inserted into the output frame in the order they were placed in the original mosaic with pixels in the most recently placed subrasters replacing those in earlier placed ones in the overlap regions.Undefined pixels in the output frame are given the value stored in the keyword CCD_NUL.

The position of the reference subraster in the output frame may be adjusted by setting the offset parameters xref and yref. The edges of each subraster may be trimmed before insertion into the output frame by setting the CCD keyword MO_TRIM.

**See also:** SET/CCD, SHOW/CCD, CREATE/MOSAIC, SHIFT/MOSAIC, ALIGN/MOSAIC, FIT/MOSAIC

**Examples:**
```
MATCH/MOSAIC mosin mosout mosdb coords,coordtab 6,5
```
Align and match the mosin frame with respect to subraster 6, 5

```
SET/CCD MO_TRIM=2,2,2,2
```
MATCH/MOSAIC mosin mosout mosdb coords,coordtab 6,5 The same as above but trim 2 rows and columns off of each input image before inserting into the output frame.

# MKREDT/CCD

| MKREDT/CCD | *stdred/ccdred* | 1-Nov-1993 RHW |
|---|---|---|

**Purpose:** Create an empty CCD table containing columns for the science frames and the master calibration frames to be used in the reduction

**Syntax:** `MKRED/CCD out_tab`

`red_tab` output CCD table with empty columns that can be filled with the names of the unreduced science frames and the master calibration frames assocoated with the science frames and to be used for their reduction. The names of the columns are take from the keywords SC_COL to contain the science frames; BS_COL to contain the master bias frames; DK_COL for the master dark frames; FF_COL for the master flat frames, and SK_COL for the master sky frames. Ill defined columns (e.g. " " or "?") will not be created. No default.

**Note:** This command enables the user to create a table similar to the format of the Data Organizer association table. The table can be filled with the names of the science frames to be reduced and their associated master calibration frames.

The table can be used for the command COMBINE/CCD to create the master calibration frames, provided the naming convention is followed, or the command REDUCE/CCD to reduce the science frame(s) in the table using the calibration frames listed in the various columns.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, READ/TABLE, EDIT/TABLE, SELECT/TABLE

**Examples:** `MKRED/CCD dataorg`
Create a new and empty dataorg table with the columns defined by the keywords "exp"_COL.

## OVERSCAN/CCD *stdred/ccdred* 25-Mar-1993 RHW

**Purpose:** Correct the input frame for the bias offset determined from the overscan region region

**Syntax:** `OVERSCAN/CCD [in_fram] [out_fram] [sc_area] [direct] [mode]`

| | |
|---|---|
| `in_fram` | input frame to be corrected. Input frame can either be a sky frame or a science frame. To determine the exposure type of the input frame it should have a valid descriptor containing a valid exposure type. Default is the name stored in the keyword CCD_IN; see below for details. |
| `out_fram` | output frame resulting from subtracting the bias frame from the input frame. No default. |
| `sc_area` | overscan area. Default is the area defined in the keyword OV_SEC. |
| `direct` | readout direction of the CCD: 'column' or 'row'; default is the setting stored in the CCD keyword DIRECT. |
| `mode` | mode of operation, interactive (I) or automatic (A). Default is the current setting (keyword OV_IMODE) |

**Note:** Depending on the keyword DIRECT, the command averages the rows in the overscan region (DIRECT="col") or averages the columns (DIRECT="row").

The area over which the average is calculated is obtained from the CCD keyword OV_SEC. In case the keyword DIRECT is not filled with 'row' or 'col', the average is taken row-wise if the area is large in y than in x; else the average is taken column-wise.

The command can correct all types of frames, including a bias itself. The default mode or operation is taken from the keyword OV_IMODE. Except for the overscan area itself, all fit parameters are taken from the keyword setting. These keywords and their meaning are:
OV_FUNCT lin/pol - linear or polynomial function;
OV_ORDER numb - the order of the polynomial function;
OV_AVER numb - box smooth with a width of 'number' pixels;
OV_ITER numb - maximum number of iterations;
OV_REJEC numb - sigma rejection factor.

After the overscan average is made the result is fitted, either automatically, or interactively using the graphics cursor. In case of interactive overscan fitting the command will first use the current keyword setting. The result will be displayed on the graphics window. The cursor will appear to select the range to be fitted. The fit will be overplotted and the user will be asked if this is satisfactory. In case of 'N' all fitted parameters can be modified and a new fit loop will start.

After the overscan vector is determined an overscan frame is created. This frame will be used to correct the input frame for the overscan bias. In case the overscan correction is a constant (zero order approximation: the mean of the offset in the overscan region) this constant will be used.

The command will not try to correct for the overscan offset if that has already been done. The descriptor CCDSTAT in the output frame is updated to indicate that the frame has been trimmed.

**See also:** SET/CCD, SHOW/CCD, REDUCE/CCD, REGRESSION/LINEAR, REGRESSION/POLYNOMIAL

**Examples:**
```
OVERSCAN/CCD susi0097 susi0097sccor ?  col
```

# OVERSCAN/CCD

Average the overscan columns using the keyword setting OV_xxx, and the overscan area defined in the keyword OV_SEC. The readout direction is column wise. Subtract the overscan fit from the susi0097 to obtain the bias correction output frame susi0097sccor.

REDUCE/CCD *stdred/ccdred* 25-Mar-1993 RHW

**Purpose:** Do the (partial) calibration of one or more frames

**Syntax:** `REDUCE/CCD [in_spec] [out_frm] [o_flag]`

| | |
|---|---|
| `in_spec` | single (science) frame or MIDAS CCD input table (with the extension .tbl) containing the names of the frames and the names of the master calibration frame(s) to be used for the calibration. Default is the input specification stored in the CCD keyword CCD_IN. |
| `out_frm` | name of the (partially) reduced science frame. The default name is the name stored in the keyword CCD_OUT is used. If that is empty the output frame name is the same as the input with the extension "_reduc". NOTE: This parameter is only used is case of single frame input for the first parameter. |
| `o_flag` | Overwrite flag. If "Y" existing reduced will be overwritten. Default N. |

# REDUCE/CCD

**Note:** The command can be used to reduce a single frame or a list of frames stored in a MIDAS input table. The default input is taken from the keyword CCD_IN.

First, the case of a single frame reduction:
In this case the first parameter should contain the name of the frame (without the extension .bdf). The command assumes that the name of the calibration frames are stored in the keyword SC_expFRM, where 'exp' is the two letter code for the exposure type, AND that these calibration frame are available.
The default output name is the identical to the name of the input frame with the extension "_reduc".

In the case multiple frame reductions:
In this case the command expects a MIDAS table as input, including the extension .tbl. The command will take all information from this table. i.e. the input science frames, and the names of the master calibration frames to be used. The name of the output science frame is the identical to the name of the input frame with the extension "_reduc"

This is what the command will do to your data:
1. The command will look in the SC_ keywords (see below) to determine what corrections are to be applied (for the keywords, see below);
2. As far as the master calibration frames concern, it checks for the existence of these frames.
In the case of table input, if a calibration frame of a particular exposure type is required but not available, the calibration process will be suspended, and the command will first try the create that calibration frame, assuming the CCD naming convention and using the command COMBINE/CCD. Thereafter, the calibration will continue.
In case of single frame input, if the calibration frame is not available this calibratiob step will not be executed. 3. By reading the CCD descriptor of the science and calibration frames the command checks if a calibration step was already done. If so, this step will not be executed, even if it was asked for.
4. After all calibration frames are present it will, in a sequential order do the requested calibration (if not done already, see below). The following steps can be executed:
correct for overscan (keyword SC_SCAN);
trimming (keyword SC_TRIM)
bad pixel correction (keyword SC_FXPIX);
bias correction (keywords SC_BSCOR and SC_BSFRM);
dark correction (keywords SC_DKCOR and SC_DKCOR);
flat fielding (keywords SC_FFCOR and SC_FFCOR);
illumination correction (keywords SC_ILCOR and SC_ILFRM);
fringing correction (keywords SC_FRCOR and SC_FRFRM).
5. After a calibration step has finished successfully the command will write a flag into the frame descriptor, indicating that the calibration step(s) was (were) completed.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD

**Examples:** `REDUCE/CCD rost_asso.tbl`
Start the ccd reduction sequence. Use the reduction table redtbl.tbl. All calibration frames will be obtained from that reduction table. Do the calibration according to the keyword settings.

`REDUCE/CCD m100rraw ?`

Start the ccd reduction sequence for the frame m100rraw. The output frame will be m100raw_reduc.
The command will read the keywords SC_xxCOR to find out which calibration are wanted. The names of the calibration frame are taken from the keywords SC_xxFRM.

`REDUCE/CCD`
Start the ccd reduction sequence. Use the reduction table stored in the keyword CCD_IN. Use the settings of the keywords SC_xxCOR to get the calibration options.

---

## SAVE/CCD *stdred/ccdred* 1-Nov-93 RHW

**Purpose:** Save the current CCD keywords and control table into a session table.

**Syntax:** `SAVE/CCD name`

`name` session name; no default

**Note:** The command will save the CCD keywords as well as the CCD reduction table into a table called P1_REDUC.tbl, where P1 is the session name you gave as input. In case the reduction table doesn't exist an empty table will be created.

The command 'INIT/CCD name' can be used to restore the session status.

**See also:** INIT/CCD, LOAD/CCD

**Examples:** `SAVE/CCD mysession`
Save the CCD keyword structure and and the reduction table stored in the table mysession_REDUC. If no reduction CCD table exists only the CCD keyword settings will be saved in that table.

# SET/CCD

| SET/CCD | *stdred/ccdred* | 1-Nov-1993 RHW |
|---|---|---|

**Purpose:** Define the values of parameters in the current CCD session. Up to 8 keywords can be defined in one single command.

**Syntax:** `SET/CCD keyw=value [...]`

`keyw` parameter name

`value` parameter value

**Note:** No space character can be inserted on the right or left part of the sign "=". The name of the CCD keyword can be truncated to a non-ambiguous root.

Short description of CCD keywords is provided by HELP/CCD command, values are displayed by SHOW/CCD.

It is possible to assign a value to CCD keywords using monitor assignment. In this case, there must be a space on both parts of the sign =. The syntax depends on the definition of the keyword (provided by HELP/CCD).
Examples:
BS_MET = sig (keyw. BS_MET/C/1/20)
SK_WEI = yes (keyw. SK_WEI/R/1/1)

To assign values to character keywords, additional spaces must be inserted to overwrite a longer previous value.

To assign the value of an element of a multiple element keyword (e.g. IL_SIGMA/R/1/2) it is possible to use the direct assignment (IL_SIGMA(2) = 4.) or the SET/CCD by inserting commas in place of the values that must not be changed.

**See also:** SHOW/CCD, LOAD/CCD, HELP/CCD

**Examples:** `SET/CCD OBSER="ik zei de gek" BS_MET=sig IL_SIGMA=4,4`
Set the keywords 'OBSERVER', 'BS_MET' and 'IL_SIGMA' to the values given.

`SET/CCD IDENT="my_run"`
Set the IDENT keyword to 'my_run'.

## SHIFT/MOSAIC *stdred/ccdred* 02-Aug-1995

**Purpose:** Get x and y shifts of the subraster in the mosaic frame

**Syntax:** `SHIFT/MOSAIC out_tab [curs_opt] [csx,csy] [clear_opt]`

| | |
|---|---|
| `out_tab` | Output table to contain the pixel shifts in x and y of the individual subrasters in the mosaic frame. No default. |
| `curs_opt` | centering algorithm to be used for determining the position. The following options are available:<br>SC - Single Cursor, using the simple GET/CURSOR command to get the coordinates from the image display;<br>CG - for get the position using the CENTER/GAUSS command; CM - for get the position using the CENTER/MOMENT command; CI - for get the position using the CENTER/IQE command;<br>Default is SC. |
| `csz,csy` | number of x-and y-pixels for subwindow centered via the single cursor; The option is only applicable for if the curs_opt is CG, CM, or CI. Default is 50,50. |
| `clear_opt` | clear option for clearing the display overlay channel before getting the cursor positions. Default is Y. |

**Note:** In order to align and/or match the subrasters in a frame mosaic created by the command CREATE/MOSAIC, the user must give pixel shift information. The shift information can be input in a number of different ways. For more information about these possibilities see the help text of the commands ALIGN/MOSAIC and MATCH/MOSAIC.

This command store the positional informal of object in overlapping regions in adjacent subrasters in a table. Subsequently this table can be used by the commands ALIGN/MOSAIC and MATCH/MOSAIC to compute the pixel offsets in x and y and to align the subraster. The must be loaded in the display window.

In order to get the ALIGN/MOSAIC or the MATCH/MOSAIC commands work correctly the positions of the objects must be obtained in the following sequence:
1) the x and y coordinates of the object in the first subraster;
2) the x and y coordinates of the same object in the second, adjacent subraster;
3) the x and y coordinates of the next object in the first or another subraster;
4) the x and y coordinates of the same object in the second, adjacent, subraster;
etc.

The command creates an output table containing the columns :X_coordpix, :Y_coordpix, and :Value. Additional columns can be present and depends on the cursor option used.

**See also:** SET/CCD, SHOW/CCD, CREATE/MOSAIC, ALIGN/MOSAIC, MATCH/MOSAIC

**Examples:** `SHIFT/MOSAIC dorcoord CG ? N`
Get the positions of pairs of objects in adjacent subrasters in the mosaic frame loaded on the display. Stored the positions in the table "dorcoord". Use the CENTER/GAUSS command to get the positions. Don't erase the overlay channel.

# SHOW/CCD

| SHOW/CCD | *stdred/ccdred* | 1-Nov-1993 RHW |
|---|---|---|

**Purpose:** Show (part of) the setup of the CCD package

**Syntax:** `SHOW/CCD [subject]`

| | |
|---|---|
| `subject` | Section of the keyword structure to be shown. The names of the sections are (default is the complete listing):<br>GE: keywords containing general setup parameters;<br>BS: keywords for combining bias frames;<br>DK: keywords for combining dark frames;<br>FF: keywords for combining flat frames;<br>SK: keywords for combining sky frames;<br>IL: keywords for making an illumination frame;<br>FR: keywords for making a fringe frame;<br>OV: keywords for determining the overscan offset;<br>SC: keywords for science frame(s) reduction;<br>HE: general for the CCD context.<br><br>Default is GE. |

**Note:** None

**See also:** HELP/CCD, SET/CCD

**Examples:** `SHOW/CCD BS`
Show the keywords and their values relevant for combining the bias frames into a master bias.

SKYCOR/CCD *stdred/ccdred* 25-Mar-1993 RHW

**Purpose:** Make sky illumination correction frame(s)

**Syntax:** `SKYCOR/CCD [in_spec] [out_frm] [xboxmn,xboxmx] [yboxmn,yboxmx] [clip] [lowsig,higsig]`

`in_spec` single flat frame or MIDAS CCD association table (with the extension .tbl) containing the names of the flat frame(s) in the column :SKY. Default input is taken from the keyword CCD_IN.

`out_frm` name of the output illumination frame. Default output is the name of the input table with extension "_ill". Only used in case of single frame input for the first parameter. In case the input is a MIDAS table the name(s) of the output frame(s) will have the same name as the sky frame field except for the extension '_ill' (see below).

`xboxmn,xboxmx` minimum and maximum smoothing box size along the x axes. The minimum box size is used at the edges and grows to the maximum size in the middle of the image. This allows the smoothed image to better represent gradients at the edge of the image. If a size is less then 1 it is interpreted as a fraction of the image size. If a size is greater than or equal to 1 then it is the box size in pixels. A size greater than the size of image selects a box equal to the size of the image. Default values are taken from the keyword IL_XBOX.

`yboxmn,yboxmx` see above. Default values are taken from IL_YBOX.

`clip` Clean the input frame(s) of objects? If yes then a clipping algorithm is used to detect and exclude objects from the smoothing. Default taken from IL_CLIP.

`lowsig,higsig` sigma clipping thresholds above and below the smoothed illumination. Default values are taken from the keyword IL_SIGMA. The keyword is only read in case clip=yes.

# SKYCOR/CCD

**Note:** In case the input is an association table the command will check for the existence of the sky frames. Sky frames that do not exit will be created first, provided the association table contain the sky column. Other relevant calibration frames for processing the sky frames, including flat fields will be taken from the corresponding calibration columns belonging to the same science frame. Output illumination frame(s) will have the same name as the original sky frame expanded with '_ill'.

In the case of single frame input and all calibration frames input will be taken from the SC_ keywords and must exist.

The command operates very similar to the command REDUCE/CCD, which the exception that the command not check for the exposure type of the input frame. Master frames to be used to calibrate the input frame first are taken for the SC_ keywords (in case of single frame input), or from the reduction table in case of table input. For details see the command REDUCE/CCD.

The input frame are automatically processed, including flat field, before computing the illumination. The input frame are generally blank sky calibration frames which have the same illumination and instrumental effects as the object observations. Science frames may be used but removal of the objects may not be very good; particularly large, bright objects.

The illumination frame is produced by heavily smoothing the calibrated input frames using a moving "boxcar" average. The effects of objects in the frames may be minimized by using a sigma clipping algorithm to detect and exclude the objects from the average. The output illumination frame can used by REDUCE/CCD to remove the illumination pattern in the science frames.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD, REDUCE/CCD, SKYFLAT/CCD

**Examples:** `SKYCOR/CCD ccd_asstbl.tbl`
Reduce all sky frames (including flat fielding) in the the table ccd_asstbl.tbl in the column :SKY. All master calibration frames will be obtained from the same table. Do the calibration according to the keyword settings. After the calibrations are done create the illumination frames, using the 'IL_' keyword settings.

`SKYCOR/CCD m100rraw_sk m100rraw_il`
Start the ccd reduction sequence for the frame m100rraw_sk. The output illumination frame will be m100rraw_il. In case only dark and flat field correction is to be applied the names of these two calibration frames will be obtained from the keyword SC_DKFRM and SC_FFFRM, respectively.

SKYFLAT/CCD *stdred/ccdred* 25-Mar-1993 RHW

**Purpose:** Apply sky observation to flat field to remove illumination pattern

**Syntax:** `SKYFLAT/CCD [in_spec] [out_frm] [xboxmn,xboxmx] [yboxmn,yboxmx]`
`[clip] [lowsig,higsig]`

`in_spec` single sky frame or MIDAS CCD reduction table (with the extension .tbl) containing the names of the sky frame(s) in column :SKY. Default input is taken from the keyword CCD_IN.

`out_frm` name of the illumination corrected sky flat field. Default the output file the same as the input sky flat frame. Hence, by default the input sky flat frame will be OVERWRITTEN.
This parameter is only used is case of single frame input for the first parameter. In case the output is taken a reduction table the output frame(s) will also OVERWRITE the original flat frames.

`xboxmn,xboxmx` minimum and maximum smoothing box size along the x axes. The minimum box size is used at the edges and grows to the maximum size in the middle of the image. This allows the smoothed image to better represent gradients at the edge of the image. If a size is less then 1 it is interpreted as a fraction of the image size. If a size is greater than or equal to 1 then it is the box size in pixels. A size greater than the size of image selects a box equal to the size of the image. Default values are taken from the keyword IL_XBOX.

`yboxmn,yboxmx` see above. Default values are taken from IL_YBOX.

`clip` Clean the input frame(s) of objects? If yes then a clipping algorithm is used to detect and exclude objects from the smoothing. Default taken from IL_CLIP.

`lowsig,higsig` sigma clipping thresholds above and below the smoothed illumination. Default values are taken from the keyword IL_SIGMA. The keyword is only read in case clip=yes.

# SKYFLAT/CCD

**Note:** The command operates very similar to the command REDUCE/CCD, which the exception that the command not check for the exposure type of the input frame.

In case the input is an association table the command will check for the existence of the flat frames in the column :SKY in the association table. Flat frames that do not exit will be created first, provided the association table contain the :SKY column. Other relevant calibration frames for processing the flat frames, will be taken from the corresponding calibration columns.

After the reduced sky frame is obtained, the sky illumination frame is produced by heavily smoothing the calibrated sky frames using a moving "boxcar" average. The effects of objects in the frames may be minimized by using a sigma clipping algorithm to detect and exclude the objects from the average. This illumination frame will be used to make an illumination corrected flat field (sky flat) according to the formula:
f_new = f_old*ill_sky/scale, where
f_new is the corrected flat field, f_old is the original input flat, used to reduce the blank sky frame, ill_sky is the illumination correction determined from the sky, and scale is the nomalisation factor.

The original flat frame is multiplied by the scaled sky illumination correction frame to produce the illumination corrected flat. The output will OVERWRITE the original (reduced) flat frames. Hence, the output sky flat will automatically be used as flat in subsequent calibration procedure(s).
NOTE: THE MASTER FLAT FIELDS WILL BE OVERWRITTEN BY THE CORRECTED ONES.
In the case of single frame input and all calibration frames input will be taken from the SC_ keywords and must exist. Also here, by default in the output illumination corrected flat will replace the input flat.

For more information see the CCD chapter in Volume B of the User's manual.

**See also:** SET/CCD, SHOW/CCD, HELP/CCD, REDUCE/CCD, SKYCOR/CCD,

**Examples:** `SKYFLAT/CCD redtbl.tbl`
Reduce all sky frames in the the ccd reduction table redtbl.tbl. All master calibration frames will be obtained from that reduction table. Do the calibration according to the keyword settings, including the flat fielding. From the calibrated sky produce the illumination frame, using the 'IL_' keyword settings. Hereafter compute the output sky flat by multiplying the illumination frame with the original flat field. The output sky flats will have overwrite the original flats.

`SKYFLAT/CCD m100rraw_sk m100rraw_sf`
Start the ccd reduction sequence for the frame m100rraw. The output frame will be m100rraw_sf. In case only dark and flat field correction is to be applied the names of these two calibration frames will be obtained from the keyword SC_DKFRM and SC_FFFRM, respectively.

TRIM/CCD *stdred/ccdred* 25-Mar-1993 RHW

**Purpose:** Extract the useful data from the ccd frame.

**Syntax:** `TRIM/CCD [in_fram] [out_fram] [im_sec] [del_flg]`

| | |
|---|---|
| `in_fram` | input frame to be trimmed; default is the name stored in the keyword CCD_IN. |
| `out_frame` | resulting output frame; no default. |
| | im_sec = image section to be extracted. Default is the section stored in the keyword IM_SEC. |
| `del_flg` | delete flag. Default is 'NO'. |

**Note:** The command will not try to trim the input frame if that has already been done. The descriptor CCDSTAT in the output frame is updated to indicate that the frame has been trimmed.

**See also:** HELP/CCD, REDUCE/CCD, EXTRACT/IMAGE

**Examples:** `TRIM/CCD fieldin fieldext ? y`
The input frame fiedlin will be trimmed. The output frame fieldextr will contain the output area, stored in the keyword IM_SEC. After extraction the input frame will be deleted.

# Appendix F

# Standard Reduction - ccdtest

# Standard Reduction Commands

# Context: ccdtest

# TESTB1/CCD

| TESTB1/CCD | *stdred/ccdtest* | 20-Jun-1994 RHW |
|---|---|---|

**Purpose:** Combine bias frames stored in a catalogue and display it

**Syntax:** `TESTB1/CCD in_cat [out_id] [meth] [option]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `meth` | combining method for creation of the master bias frame. Possible methods are:<br>sum - simple sum of all input input images;<br>ave(rage) - combined by averaging;<br>median - combining by medianing each pixel;<br>min(reject) - reject the min value from the average;<br>max(reject) - reject the max value from the average;<br>minmax(reject) - reject min and max from the average;<br>sig(clip) - sigma clipping to each pixel;<br>avsig(clip) - sigma clipping with minmax rejection;<br><br>Default is the average sigma cliping method. In case a clipping algoritms is used the CCD keyword BS_CLP will be read to get the clipping factors. |
| `option` | Option to enable overwriting of existing combined bias frame. Default is 'N' (i.e. no overwrite). |

**Note:** This command is the first or a series of test commands on bias frames: B1 to B5. The complete test is executed by the command TESTBA/CCD.
To avoid unnecessary computations the command checks for the presence of the combined bias. If the overwrite option 'Y' is used the existing combined bias output frame will be overwritten.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.bdf: the combined bias frame;
'out_id'.ps: postscript file with an image of the combined bias frame;

Keywords BIASMEAN, BIASSIGM are filled with the mean bias and its standard deviation.

**See also:** TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTB1/CCD bs_cat.cat BS median`
Run the test procedure on the frame in the catalogue bs_cat. Combine the frame in a master frame BS using a median filter. All output will start with the identifier BS.

TESTB2/CCD *stdred/ccdtest* 20-Jun-1994 RHW

**Purpose:** Compute row and column average of a (averaged) bias frame

**Syntax:** `TESTB2/CCD in_frm [out_id] [row_ran] [col_ran]`

| | |
|---|---|
| `in_frm` | input frame; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `row_ran` | Range of rows in the master bias frames to be averaged. Default is all rows. |
| `col_ran` | Range of columns in the master bias frames to be averaged. Default is all columns. |

**Note:** This command is the second of a series of tests on BIAS frames: B1 to B5. See the command TESTBA/CCD.
From the combined bias frame rows and columns are averaged and plotted.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'_avcol.bdf: MIDAS image containing the column average;
'out_id'_avrow.bdf: MIDAS image containing the row average;
'out_id'_aver.ps: postscript file with the plots of the column and row average;

**See also:** TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTB2/CCD BIAS BS median ? 350,500`
Run the test on the frame BIAS. Do averaging over all rows and over the column range from 350 to 500.

# TESTB3/CCD

| TESTB3/CCD | *stdred/ccdtest* | 20-Jun-1994 RHW |
|---|---|---|

**Purpose:** Find the hot pixels in a (combined) bias frame

**Syntax:** `TESTB3/CCD in_frm [out_id] [area] [size] [option]`

| | |
|---|---|
| `in_frm` | input frame; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `area` | area which the hot pixels will be determined. Reducing the area will increase the speed of the execution. Default is whole image area. |
| `size` | size of smoothing box used for median filtering used for obtaining the hot pixel list. Default is 5. |

**Note:** This command is the third test in a series of tests on bias frames: B1 to B5. See the command TESTBA/CCD.
The command will try to find the hot pixels. The combined bias frame is median filtered (using the parameter 'size') and subtracted from the original. A plot is generated showing the position of the hot pixels and the affected columns. Hot pixels will only be searched for within the requested area and above the intensity level of (mean + 0.25*sigma + 5.), where mean is the mean intensity level, sigma is the standard deviation.
To avoid unnecessary computations the command checks for the presence of the median filtered hot pixel frame and does not recompute this frames if is already present.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'_hotpix.tbl: MIDAS table containing the hot pixels;
'out_id'_hotpix.ps: postscript file with a plot of the hot pixels

**See also:** TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTB3/CCD BIAS BS [@900,@900:@1100,@1100]`
Run the test procedure on the frame BIAS. All output will start with the identifier BS. Use the default for median filter box.

TESTB4/CCD *stdred/ccdtest* 20-Jun-1994 RHW

**Purpose:** Make a histogram of the pixel intensities and rebin the input frame

**Syntax:** `TESTB4/CCD in_frm [out_id] [area] [size,fac]`

| | |
|---|---|
| `in_frm` | input frame; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `area` | area which the histogram is computed; Default is whole image area. |
| `size,fact` | size of smoothing box used for median filtering and decimation factor for frame size reduction. Default is 5,16. The default size 5 for the filtering implies a filtering over boxes of 11 by 11 pixels. |

**Note:** This command is the fourth test of a series of tests bias frames: B1 to B5. See the command TESTBA/CCD.
The command will first correct the frame for hot pixels and then rebin it. A histogram of this rebinned frame is made.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'_hist.ps: postscript file with a plot of the histogram;
'out_id'_rebin.bdf: cleaned and rebinned combined bias frame;
'out_id'_rebin.ps: postscript file with an image of the above;

**See also:** TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTB4/CCD BIAS BS [@900,@900:@1100,@1100]`
Run the test procedure on the frame BIAS. Use the defaults for the rebinning.

# TESTB5/CCD

| TESTB5/CCD | *stdred/ccdtest* | 20-Jun-1994 RHW |
|---|---|---|

**Purpose:** Do the statistics of the bias frame is a catalogue.

**Syntax:** `TESTB5/CCD in_cat [out_id] [area] [size,fac]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `area` | area which the statistics is computed; Default is whole image area. |
| `size,fact` | size of smoothing box used for median filtering and decimation factor for frame size reduction. Default is 5,16. The default size 5 for the filtering implies a filtering over boxes of 11 by 11 pixels. |

**Note:** This command is the fifth test of a series of tests bias frames: B1 to B5. See the command TESTBA/CCD.
For each input frame in the catalogue determine the mean BIAS and standard deviation AFTER hot pixel correction, box averaging and median filtering. Finally, the overall mean bias level and sigma are computed.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The statistical data wil be stored in the table 'out_id'_mean.tbl.

**See also:** TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTB5/CCD bs_cat.cat BS`
Do the statistics on all frame in the catalogue bs_cat. Use the defaults for filtering and rebinning.

TESTBA/CCD *stdred/ccdtest* 20-Jun-1994 RHW

**Purpose:** Do a series of tests of a catalogue of bias frames

**Syntax:** `TESTBA/CCD in_cat [out_id] [meth] [row_ran] [col_ran] [area] [size,fac]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is BIAS. |
| `meth` | combining method for creation of the master bias frame. Possible methods are:<br>sum - simple sum of all input input images;<br>ave(rage) - combined by averaging;<br>median - combining by medianing each pixel;<br>min(reject) - reject the min value from the average;<br>max(reject) - reject the max value from the average;<br>minmax(reject) - reject min and max from the average;<br>sig(clip) - sigma clipping to each pixel;<br>avsig(clip) - sigma clipping with minmax rejection;<br><br>Default is the average sigma cliping method. In case a clipping algoritms is used the CCD keyword BS_CLP will be read to get the clipping factors. |
| `row_ran` | Range of rows in the master bias frames to be averaged. Default is all rows. |
| `col_ran` | Range of columns in the master bias frames to be averaged. Default is all columns. |
| `area` | area which the hot pixels will be determined. Reducing the area will increase the speed of the execution. Default is whole image area. |
| `size,fact` | size of smoothing box used for median filtering and decimation factor for frame size reduction. Default is 5,16. Median filtering is applied on the master bias frame for obtaining the hot pixels. The efault size 5 for the filtering implies a filtering over boxes of 11 by 11 pixels. |

# TESTBA/CCD

**Note:** This command should be the first command to be executed for a complete test of the CCD characteristics, and should be followed by the command TESTFA/CCD (test of low count flat fields).

The command will do the following tests on the frame in the input catalogue:
Test B1: Creation of the combined bias frame. The result is loaded onto the display.
Test B2: Find the hot pixels. The combined bias frame is median filtered (using the parameter 'fil_siz') and subtracted from the original. A plot is generated showing the position of the hot pixels and the affected columns. Hot pixels will only be searched for within the requested area and above the intensity level of (mean + 0.25*sigma + 5.), where mean is the mean int level, simga is the standard deviation.
Test B3: Inspection of single frames. From the combined bias frame rows and columns are averaged and plotted.
Test B4: The last frame in the catalogue is first corrected for hot pixels and then rebinned. A histogram of this rebinned frame is made.
Test B5: For each input frame in the catalogue determine the mean bias and standard deviation after hot pixel correction (using the hot pixel table determined in test B2), box averaging and median filtering. The keyword BIASMEAN and BIASSIGM are filled with the average values for the mean and sigma.
To avoid unnecessary computations the command checks for the presence of the combined bias frame and the median filtered hot pixel frame and does not recompute these frames if they are already present. In the case of subtests the command will (re)created these output frames.
Single tests on the bias frames can be executed using the commands TESTBn/CCD with n = 1,2,3,4 or 5.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.bdf: the combined bias frame;
'out_id'.ps: postscript file with an image of the combined bias frame;
'out_id'_hotpix.tbl: MIDAS table containing the hot pixels;
'out_id'_hotpix.ps: postscript file with a plot of the hot pixels
'out_id'_avcol.bdf: MIDAS image containing the column average;
'out_id'_avrow.bdf: MIDAS image containing the row average;
'out_id'_aver.ps: postscript file with the plots of the column and row average;
'out_id'_hist.ps: postscript file with a plot of the histogram;
'out_id'_rebin.bdf: cleaned and rebinned combined bias frame;
'out_id'_rebin.ps: postscript file with an image of the above;
'out_id'_mean.tbl: MIDAS table with the mean and standard deviation valuesb of the single bias frames.
The keywords BIASMEAN, BIASSIGM are filled with the mean bias and its standard deviation.

**See also:** TESTBn/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, AVERAGE/ROW, AVERAGE/COLUMN FILTER/MEDIAN, Ch. 3 in Vol. 2 of the Users Guide.

**Examples:** `TESTBA/CCD bs_cat.cat BS median ?  350,500 [@900,@900:@1100,@1100]`
Run the test procedure on the frame in the catalogue bs_cat. Combine the frame in a master frame BS using a median filter. All output will start with the identifier BS. Do averaging over all rows and over the column range from 350 to 500. Use the defaults for median filter box and decimation factor.

TESTC/CCD *stdred/ccdtest* 21-Jun-1994 RHW

**Purpose:** Compute the horizontal and vertical charge transfer efficiency.

**Syntax:** `TESTC/CCD in_frm [rows] x_pix [colums] y_pix`

| | |
|---|---|
| `in_frm` | input flat frame. The frame should have an illumination level as high as possible, however with being saturated |
| `rows` | the range in y (rows) to be averaged into a single line and to be used for computing the HCTE. Default all rows. In case of 0.0 as input the HCTE will not be computed. |
| `x_pix` | in the x direction: first and last pixel of the image section on the CCD, and the coordinate for the first bias overscan pixel (all in pixel coordinates). No default. |
| `columns` | the range in x (columns) to be averaged into a single line and to be used for computing the VCTE. Default all columns. In case of 0,0 as input the VCTE will not be computed. |
| `y_pix` | in the y direction: first and last pixel of the image section on the CCD, and the coordinate for the first bias overscan pixel (all in pixel coordinates). No default. |

**Note:** For the HCTE the command first averages the rows given as the second parameter. From the x_range parameter the number of image pixels, the last image pixel and the first bias overscan pixel is obtained and computes the HCTE according to the formula:
HCTE = 1 - bc/ic*ni, where:
- bs are the counts above the bias level in the first overscan pixel in a row;
- ic are the count above the bias level in the last image pixel in a row;
- ni are the number of image pixels in a row.

The values for the bias offset is extracted from the keyword BIASMEAN, and in computed by the command TESTB/CCD.

To determine the image section of the CCD and the overscan regions one can use the commands READ/IMAGE, PLOT/COLUMN and PLOT/ROW.

Note that the last column of a row is often slightly brighter than the rest of the row (because the pixel is slightly larger).

The vertical charge transfer efficiency is computed in a similar way.

**See also:** TESTB/CCD, READ/IMAGE, PLOT/COLUMN, PLOT/ROW

**Examples:**
```
testc/ccd red0804 <,> 1,2067,2068 0,0
```
Determine the HCTE from the frame red0804. The command will average all rows. The HCTE is computed using pixel 2068 as the first bias overscan pixel and pixel 2067 as the last image pixel. There are 2048 valied image pixels in the x direction. The VCTE is not computed.

# TESTD/CCD

| TESTD/CCD | *stdred/ccdtest* | 21-Jun-1994 RHW |
|---|---|---|

**Purpose:** Do a test on a catalogue of dark current frames

**Syntax:** `TESTD/CCD in_cat [out_id] [dec_fac]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined; no default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is DARK. |
| `dec_fac` | decimation factor for frame size reduction. Default is 16. |

**Note:** This command requires the keyword BIASMEAN to be filled and hence should be executed after the command TESTB/CCD.
The command does the following test on the frames in the input catalogue.
The input frames in the catalogue will be corrected for the bias offset stored in the keyword BIASMEAN, normalized to one hour exposure time and multiplied by the electron/ADU conversion factor (computed by the command TESTT/CCD and stored in the keyword ADUCF). The output image is then resampled where each pixel of the rebinned frame is the median of a dec_fac x dec_fac pixels area of the input frame. The mean values and standard deviations of the rebinned frames are displayed.

The command produces the following output information:
'out_id'.bdf: rebin of the combined dark frames in units of e-/pix/hour" 'out_id'.ps: postscript file of the above" 'out_id'_cont.ps contour plot of the above (same units)"
Instead of using a single number to correct for the bias, a mean bias frame, e.g. the one created by the command TESTB/CCD can be used. For that fill the keyword BIASMEAN with the name of that bias frame.

**Output:** The keyword

**See also:** SET/CCD, SHOW/CCD, TESTB/CCD, TESTT/CCD

**Examples:** `TESTD/CCD dk_cat MYDARK 20`
Run the test procedure on the frames in the catalogue dk_cat. Combine all rebined frames into a frame MYDARK. Files for the image and contour plot are MYDARK.ps and MYDARK_cont.ps respectively.

TESTF1/CCD *stdred/ccdtest* 20-Jun-1994 RHW

**Purpose:** Combine the flat frame in the input catalogue and display

**Syntax:** `TESTF1/CCD in_cat [out_id] [meth] [area] [exp_ran] [option]`

`in_cat` input (with the extension .cat) catalogue of frames to be combined; no default.

`out_id` output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is FLAT.

`meth` combining method for creation of the master low-count- level flat. Possible methods are: sum - simple sum of all input input images;
ave(rage) - combined by averaging;
median - combining by medianing each pixel;
min(reject) - reject the min value from the average;
max(reject) - reject the max value from the average;
minmax(reject) - reject min and max from the average;
sig(clip) - sigma clipping to each pixel;
avsig(clip) - sigma clipping with minmax rejection;

Default is the average sigma clipping method. In case clipping algoritms are used the keyword FF_CLP will be read to get the clipping factors.

`area` area of the input frames in which the pixels will be considered. Default is the whole image area.

`exp_ran` Range of exposure times to be combined into the master flat field. The exposure time is normally recored in the descriptor O_TIME(7). Default 0,5 seconds

`option` Option to enable overwriting of existing combined flat frame. Default is 'N' (i.e. no overwrite).

**Note:** The command creation of the combined flat frame, using only those flat frames in the input catalogue that have exposure times falling within the allowed range. The combined flat is corrected for the bias offset. The bias offset is taken from the keyword BIASMEAN filled by the command TESTBA/CCD or TESTB5/CCD. The combined flat is loaded on the display.
The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.bdf: the combined low count flat;
'out_id'.ps: postscript file with an image of the combined low count flat.
The command checks for the presence of the combined flat frame and does not recompute the frame if is is already present.
The full test on the low count flat fields can be executed using the commands TESTFA/CCD.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**See also:** TESTFA/CCD, TESTBA/CCD, SET/CCD, SHOW/CCD, COMBINE/CCD, FILTER/MEDIAN, FIND/PIXEL

**Examples:** `TESTF1/CCD ff_cat.cat FMASTER median 5,20`
Combine all frames with exposure times between 5 and 20 seconds using the median algorithm. Subtract the bias offset.

# TESTF2/CCD

| TESTF2/CCD | *stdred/ccdtest* | 20-Jun-1994 RHW |
|---|---|---|

**Purpose:** Find the cold pixels in the combined low count flat.

**Syntax:** `TESTFA/CCD in_frm [out_id] [area] [thesh] [option]`

| | |
|---|---|
| `in_frm` | input low count flat field, possibly created using the command TESTF1/CCD. No default. |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is FLAT. |
| `area` | area of the input frames in which the pixels will be considered. Default is the whole image area. |
| `thresh` | fraction of the median count rate in the master flat frame below which intensities are considered to be generated from cold pixels on the detector. Default is 0.20. |
| `option` | Option to enable overwriting the existing median filtering input frame. Default is 'N' (i.e. no overwrite). |

**Note:** The command will list all pixels in the stacked master flat frame that show values less than thresh times the median counts in the frame. Only pixels within the area contained in 'area' are considered and repetitions of cold pixels in the increasing y coordinate are not listed.
The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'_coldpix.tbl: MIDAS table containing the cold pixels;
'out_id'_coldpix.txt: a ascii tablke containing the cold pixels.

The full test on the low count flat field is done by the command TESTFA/CCD.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**See also:** TESTFA/CCD, SET/CCD, SHOW/CCD, FILTER/MEDIAN, FIND/PIXEL

**Examples:** `TESTF2/CCD FMASTER FMASTER ? 0.20 Y`
Find all cold pixels with less than 20 percent of the median pixel value in the combine flat. Is the input frame was already median filtered overwrite the filtered frame and create a new one.

TESTFA/CCD *stdred/ccdtest* 20-Jun-1994 RHW

**Purpose:** Do a series of tests on a catalogue of low count flat frames

**Syntax:** `TESTFA/CCD in_cat [out_id] [meth] [area] [exp_ran] [theshold]`

`in_cat` input (with the extension .cat) catalogue of frames to be combined; no default.

`out_id` output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is FLAT.

`meth` combining method for creation of the master low-count- level flat. Possible methods are: sum - simple sum of all input input images;
ave(rage) - combined by averaging;
median - combining by medianing each pixel;
min(reject) - reject the min value from the average;
max(reject) - reject the max value from the average;
minmax(reject) - reject min and max from the average;
sig(clip) - sigma clipping to each pixel;
avsig(clip) - sigma clipping with minmax rejection;

In case clipping algoritms are used the keyword BS_CLP will be read to get the clipping factors.

Default is the average sigma clipping method. In case clipping algoritms are used the keyword FF_CLP will be read to get the clipping factors.

`area` area of the input frames in which the pixels will be considered. Default is the whole image area.

`exp_ran` Range of exposure times to be combined into the master flat field. The exposure time is normally recored in the descriptor O_TIME(7). Default 0,5 seconds

`thresh` fraction of the median count rate in the master flat frame below which intensities are considered to be generated from cold pixels on the detector. Default is 0.20.

# TESTFA/CCD

**Note:** This command requires the keyword BIASMEAN and BIASSIGM to be filled and hence should be executed after the commands TESTBA/CCD or TESTB5/CCD.

The command will do the following two tests on the frames in the input catalogue:
Test F1: Creation of the combined flat frame, using only those flat frames in the input catalogue that have exposure times falling within the allowed range. The combined flat is corrected for the bias offset. The bias offset is taken from the keyword BIASMEAN filled by the command TEST-B/CCD. The combined flat is loaded on the display.

Test F2: Thereafter all pixels in the stacked master flat frame that show values less than thresh times the median counts in the frame are listed. Only pixels within the area contained in 'area' are considered and repetitions of cold pixels in the increasing y coordinate are not listed.
The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.bdf: the combined low count flat;
'out_id'.ps: postscript file with an image of the combined low count flat;
'out_id'_coldpix.tbl: MIDAS table containing the cold pixels;
'out_id'_coldpix.txt: a ascii tablke containing the cold pixels.

The command checks for the presence of the combined flat frame and does not recompute the frame if is is already present.
Single tests on the flat frames can be executed using the commands TESTFn/CCD with n = 1 or 2.
For more details about the various combining methods, please refer to the CCD Chapter in Volume B of the ESO-MIDAS Users' Guide.

**See also:** TESTFn/CCD, SET/CCD, SHOW/CCD, TESTBA/CCD, COMBINE/CCD, FILTER/MEDIAN, FIND/PIXEL

**Examples:** `TESTFA/CCD ff_cat.cat FMASTER median 5,20 0.20`
Run the complete test procedure on the frames in the catalogue ff_cat. Combine all frames with exposure times between 5 and 20 seconds using the median algorithm. Subtract the bias offset and find all cold pixels with less than 20 percent of the median pxiel value in the combine flat.

TESTS/CCD *stdred/ccdtest* 21-Jun-1994 RHW

**Purpose:** Find the shutter error distribution

**Syntax:** `TESTS/CCD in_frm1 in_frm2 [out_frm] n_exp [dec_fac]`

| | |
|---|---|
| `in_frm1` | input frame containing the long flat exposure which the shutter opened and closed N times. No default. |
| `in_frm2` | input frame contained a long normal flat exposure. No default. |
| `out_frm` | output frame containing the shutter error image. Default is SHERR. |
| `n_exp` | number of short exposures in the first input frame. No default. |
| `dec_fac` | decimation factor for frame size reduction. Default is 16. |

**Note:** The command determine the accuracy of the shutter. If in_frm1 has a total reported exposure time of t1 seconds, and the shutter is opened and closed n_exp times (including the beginning and end of the exposure) and img2 has a total exposure time of t2 seconds, and the shutter is only opened and closed once, then the final shutter error frame out_frm is determined by: out_frm = (in_frm2*t1 - in_frm2*t2)/(in_frm1 - N*in_frm2).
The shutter error frame is resampled where each pixel of the output frame is the median of a dec_fac x dec_fac pixels area of the input frame.
An image and a contour plot of the error frame are stored in the files 'out_frm.bdf', 'out_frm'.ps and 'out_frm'_cont.ps respectivily.

**See also:** SET/CCD, SHOW/CCD

**Examples:** `TESTS/CCD shut1 shut2 ? 16`
Determine the shutter error from the frame shut1 and shut2. Use a median rebinning with a decimation factor 16. The default name for the shutter error frame will be used. The contour plot is stored in the frame SHERR_cont.ps; the image is in file SHERR_displ.ps.

# TESTT1/CCD

| TESTT1/CCD | *stdred/ccdtest* | 14-Jul-1994 RHW |
|---|---|---|

**Purpose:** Display the linearity and transfer curves of pairs of flat frames.

**Syntax:** `TESTT1/CCD in_cat [out_id] [area] [option]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined. The catalogues in assumed to contain a number of pairs of frames with equal exposure times. The pairs should be obtained at various exposure times. Therefore, there should be an even number (> 0) of entries in the catalogue. The first frames in each set will belong to the group "frames 1"; the second frames to group "frames 2". No default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is TRANS. |
| `area` | area of the input frame in which the data should be taken for the median and variance determinations. Default is whole image area. |
| `option` | Option to enable overwriting of existing combined bias frame. Default is 'N' (i.e. no overwrite). |

**Note:** This command requires a value for the mean bias level in the keyword BIASMEAN to be filled and hence should be executed after the command TESTBA/CCD or TESTB5/CCD. If no value or the value zero is found no bias offset will be subtracted.
The command will creation of the transfer/linearity table.
The table will contain 5 columns: column 1 for the exposure time of the first of each sets (frames 1) (label :Exp_tim1); column 2 the exposure time of the second frames (frames 2) (:Exp_tim2); column 3 the median pixel intensities over the selected frame sections in frames 1 (:Med_cnt1); column 4 the median pixel intensities over the selected area in frames 2 (:Med_cnt2) ; column 5 the variance of the difference of the frames 1 and 2 (:Variance).
To avoid unnecessary computations the command checks for the presence of the linearity table and will not created a new one if it is present, provided the option switch in 'N'.
The full linearity and transfer test is done by the command TESTTA/CCD.

**Output:** The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.tbl MIDAS table containing exposure times, median counts rates, etc, used for the analysis. 'out_id'_table.txt: table listing;
'out_id'_lin1.ps: postscript plot with lin. tests using all data;

**See also:** TESTTn/CCD, TESTBA/CCD, SET/CCD, SHOW/CCD

**Examples:** `TESTT1/CCD efosc.cat TRANS [@900,@900:@1100,@1100]`
Run the complete test procedure on frames in the catalogue efosc.cat. Create a table with all frames include and sort the table according to increasing exposure time. Then plot the exposure sequence, the linearity and the transfer curves.

TESTT2/CCD *stdred/ccdtest* 14-Jul-1994 RHW

**Purpose:** Fit the linearity cursves and determine the shutter offset

**Syntax:** `TESTT2/CCD in_tab [out_id] [select] [tim_int]`

| | |
|---|---|
| `in_tab` | input linerity table, resulting from running the test TESTT1/CCD of the test TESTTA/CCD. For the obligatory columns in the input table see the help for these commands. |
| `out_id` | output identifier for the output of the test prodedure. The output (i.e. frames, listings and plots) will start with this identifier. Default is TRANS. |
| `select` | selection criterim to include/exclude entries in the transfer/linearity table (see below) from the tests. |
| `tim_int` | interval in second within which the shutter offset will be dtermined. Default is -2,2. |

**Note:** The command will determine the linearity curve and the shutter error and the shutter offset. Entries in the linearity table not fullfilling the selection criteria select will now be selected out. From the remaining entries in the table a linear fit is done to determine the linearity curve for frames 1 and 2 and the shutter error.
Using the linearity data the fractional count rates are plotted against the median counts, appying a shutter offset in the measured exposure times. The real shutter offset is determined by the value for which the fit give the minimum mean residual.

The full linearity and transfer test is done by the command TESTTA/CCD.

**Output:** The command will store the graphics output in a the postscript file: 'out_id'_lin2.ps.

**See also:** TESTTA/CCD, SET/CCD, SHOW/CCD, SELECT/TAB

**Examples:** `TESTT2/CCD TRANS ? :Var_diff.GT.1000`
Fit the linearity data in the table TRANS.tbl. Select the data for which the variance is less than 1000. Then fit the linearity and transfer data.

# TESTT3/CCD

| TESTT3/CCD | *stdred/ccdtest* | 14-Jul-1994 RHW |
|---|---|---|

**Purpose:** Fit the transfer curve and determine the ADU conv. factor and RON

**Syntax:** `TESTT3/CCD in_tab [out_id] [select]`

| | |
|---|---|
| `in_tab` | input linearity table, resulting from running the test TESTT1/CCD or the test TESTTA/CCD. For the obligatory columns in the input table, see the help of the latter. |
| `out_id` | output identifier for the output of the test prodedure. The graphics output will start with this identifier. Default is TRANS. |
| `select` | selection criterim to include/exclude entries in the transfer/linearity table from the tests. |

**Note:** This command requires a value for the standard deviation of the bias level in keyword BIASSIGM and hence should be executed after the command TESTBA/CCD or TESTB5/CCD. If no value or the value zero is found no bias offset will be subtracted.
The command will determination of the transfer curve.
From the selected entries the table a linear regression analysis is done to determine the analog to digital conversion factor and the electronic readout noise. The readout noise is determined by the inverse of the slope between the median and the variance multiplied by the sigma of the bias (determined by TESTBA/CCD OR TESTB5/CCD and stored in keyword BIASSIGM).

The full linearity and transfer test is done by the command TESTTA/CCD.

**Output:** The command will stored the graphics output with the transfer curves in the postscript file 'out_id'_trans.ps.
The keyword ADUCF and ENOISE are filled with the ADU conversion factor and the electronic noise (in electrons/pixel).

**See also:** TESTTA/CCD, SET/CCD, SHOW/CCD, SELECT/TAB

**Examples:** `TESTT3/CCD TRANS TRANS`
Fit the linearity data in the table TRANS.tbl. Select all data. Fit the transfer curves and determine the conversion factors and RON.

TESTTA/CCD *stdred/ccdtest* 14-Jul-1994 RHW

**Purpose:** Do linearity and transfer tests on a catalogue of flat frames

**Syntax:** `TESTTA/CCD in_cat [out_id] [area] [tim_int] [select]`

| | |
|---|---|
| `in_cat` | input (with the extension .cat) catalogue of frames to be combined. The catalogues in assumed to contain a number of pairs of frames with equal exposure times. The pairs should be obtained at various exposure times. Therefore, there should be an even number (> 0) of entries in the catalogue. The first frames in each set will belong to the group "frames 1"; the second frames to group "frames 2". No default |
| `out_id` | output identifier for the output of the test prodedure. All output (i.e. frames, listings and plots) will start with this identifier. Default is TRANS. |
| `area` | area of the input frame in which the data should be taken for the median and variance determinations. Default is whole image area. |
| `tim_int` | interval in second within which the shutter offset will be dtermined. Default is -2,2. |
| `select` | selection criterim to include/exclude entries in the transfer/linearity table (see below) from the tests. |

**Note:** This command requires a value for the mean bias level in the keyword BIASMEAN to be filled and hence should be executed after the command TESTBA/CCD or TESTB5/CCD. If no value or the value zero is found no bias offset will be subtracted.
The command will do the following test on the frame in the input catalogue:
Test T1: Creation of the transfer/linearity table.
The table will contain 5 columns: column 1 for the exposure time of the first of each sets (frames 1) (label :Exp_tim1); column 2 the exposure time of the second frames (frames 2) (:Exp_tim2); column 3 the median pixel intensities over the selected frame sections in frames 1 (:Med_cnt1); column 4 the median pixel intensities over the selected area in frames 2 (:Med_cnt2) ; column 5 the variance of the difference of the frames 1 and 2 (:Variance).
Test T2: Determination of linearity curve and the shutter error and the shutter offset.
Entries in the linearity table not fullfilling the selection criteria select will now be selected out. From the remaining entries in the table a linear fit is done to determine the linearity curve for frames 1 and 2 and the shutter error.
Using the linearity data the fractional count rates are plotted against the median counts, appying a shutter offset in the measured exposure times. The real shutter offset is determined by the value for which the fit give the minimum mean residual.

Test T3: Determination of the transfer curve.
From the selected entries the table a linear regression analysis is done to determine the analog to digital conversion factor and the electronic readout noise. The readout noise is determined by the inverse of the slope between the median and the variance multiplied by the sigma of the bias (determined by TESTBA/CCD OR TESTB5/CCD and stored in keyword BIASSIGM).

To avoid unnecessary computations the command checks for the presence of the table and will not created a new one if it is present.
Single tests on the flat frames can be executed using the commands TESTTn/CCD with n = 1, 2 or 3.

# TESTTA/CCD

**Output:**

The command will stored the outputs in a number of ascii and postscript files (preceding by the identification string out_id):
'out_id'.tbl MIDAS table containing exposure times, median counts rates, etc, used for the analysis.
'out_id'_table.txt: table listing;
'out_id'_lin1.ps: postscript plot with lin. tests using all data;
'out_id'_lin2.ps: postscript plot with lin. tests using selected data;
'out_id'_trans.ps: postscript plot with transfer curves using selected data.
The keyword ADUCF and ENOISE are filled with the ADU conversion factor and the electronic noise (in electons/pixel).

**See also:** TESTTn/CCD, TESTBA/CCD, SET/CCD, SHOW/CCD, SELECT/TAB

**Examples:** `TESTTA/CCD efosc.cat TRANS [@900,@900:@1100,@1100] :Var_diff.GT.1000`
Run the complete test procedure on frames in the catalogue efosc.cat. Create a table with all frames include and sort the table according to increasing exposure time. Then start doing the analysis.

# Appendix G

# Contributed commands - daophot

# Contributed Commands

# Context: daophot

# ALLSTAR/DAOPHOT

## ALLSTAR/DAOPHOT *contrib/daophot* 22-Aug-1991 RHW

**Purpose:** Do simultaneous multiple-profile-fitting

**Syntax:** `ALLSTAR/DAOPHOT`

**Note:** Contrary to what has been documented in the Daophot II Users' manual, the maximum of pixels in a frame that can be dealt with is 1048576 which corresponds to a frame with 1024*1024 pixels. The maximum numbers of objects in a frame that can be handled is 15000.
For a complete description see the DAOPHOT II Users' manual.

**See also:** ALLSTAR/DAOPHOT, DAOMID/DAO, MIDDAO/DAO

**Examples:** `ALLSTAR/DAO`

## DAOMID/DAOPHOT *contrib/daophot* 10-May-1991 RHW

**Purpose:** DAOMID takes xxx.COO, xxx.PK, xxx.NST, or xxx.ALS and produces xxxCOO.TBL, xxxPK.TBL, xxxNST.TBL, or xxxALS.TBL respectively.

**Syntax:** `DAOMID/DAOP table`

`table` input table, in ASCII, with one of the extensions .coo, .PK, .NST or .ALS. Must adhere to DAOPHOT format.

**Note:** DAOPHOT uses pixel coordinates, and the INVENTORY and MIDAS uses both world and pixel coordinates. Hence, in converting to MIDAS, the user can give start and step size values to accomplish this transformation. Similarly, the reverse transformation can be carried out, when creating a .COO files from a .TBL file.
The user must give start and step values by writing them into a double precision keyword TRANSF, before issueing the command DAOMID/DAOP or MIDDAO/DAOP. Unless altered by the user, the keyword values will not normally change ina session.

**Examples:** `WRITE/KEY TRANSF/D/1/4 3.0,4.0,1.0,1.0`
write the double precession keyword

`DAOMID/DAOP qq.coo`
read the qq.cooo and create a midas table qqcoo.tbl

## DAOPHOT/DAOPHOT *contrib/daophot* 22-Aug-1991 RHW

**Purpose:** Do precise photometry and astrometry in a 2-dim frame

**Syntax:** `DAOPHOT/DAOPHOT`

**Note:** Contrary to what has been documented in the Daophot II Users' manual, the maximum of pixels in a frame that can be dealt with is 2560000 which corresponds to a frame with 1600*1600 pixels. The maximum numbers of objects that can be stored in the output table(s) (e.g. by the group command) is limited to 2560000/40=64000.
For a complete description see the DAOPHOT II Users' manual.

**See also:** ALLSTAR/DAOPHOT, DAOMID/DAO, MIDDAO/DAO

**Examples:** `DAOPHOT/DAOPHOT`

## MIDDAO/DAOPHOT *contrib/daophot* 10-May-1991 RHW

**Purpose:** MIDDAO takes xxx.TBL, looks for columns labelled X_COORD and Y_COORD (as produced, e.g., by cursor input), and creates the file xxx.COO (with two columns).

**Syntax:** `MIDDAO/DAOP table`

`table` input table containing (at least) two columns labelled X_COORD and Y_COORD.

**Note:** DAOPHOT uses pixel coordinates, and the INVENTORY and MIDAS uses both world and pixel coordinates. Hence, in converting to MIDAS, the user can give start and step size values to accomplish this transformation. Similarly, the reverse transformation can be carried out, when creating a .COO files from a .TBL file.
The user must give start and step values by writing them into a double precision keyword TRANSF, before issueing the command DAOMID/DAOP or MIDDAO/DAOP. Unless altered by the user, the keyword values will not normally change ina session.

**Examples:** `WRITE/KEY TRANSF/D/1/4 3.0,4.0,1.0,1.0`
write the double precession keyword

`MIDDAO/DAOP qq.coo`
read the qqcoo.tbl and create a DAOPHOTtable qq.coo

# Appendix H

# Standard Reduction - do

# Standard Reduction Commands

# Context: do

# ASSOCIATE/IMA

| ASSOCIATE/IMA | *stdred/do* | MP 05-MAY-93 |
|---|---|---|

**Purpose:** associates to scientific exposures a set of suitable calibration images

**Syntax:** `ASSOCIATE/IMA ost exptype rule_table outtable [flag] [nexp]`

| | |
|---|---|
| `ost` | Observation Summary Table |
| `exptype` | type of exposure to be associated (e.g DARK, FF, etc) |
| `rule_table` | table containing the rules for the association process |
| `outtable` | output table |
| `flag` | creation flag: (C)reate a new table or (A)ppend the new columns to an existing table. Defaulted to C |
| `nexp` | maximum number of exposures to be associated. Defaulted to 1 |

**See also:** CREATE/OST, SELECT/TABLE

**Note:** 1) The association process uses as input an Observation Summary Table created by the command CREATE/OST. This table must contain a character column labeled :EXPTYPE containing for each frame its exposure type. This column may be created by the association process (CREATE/CRULE, CLASSIFY/IMAGE)

2) The association process selects for each scientific exposure of the OST all calibration frames matching a set of user defined selection criteria . The relevant calibration frames are ranked by applying weights to each criterion and the 'nexp' best ones are selected. The search may be expanded by submitting a "second choice" set of criteria when less than nexp frames match the original one. The exposures matching the "first choice" set of criteria will be flagged with a quality flag of 1 while the images matching only the "second choice" set of criteria will be flagged with the quality 2.

3) The association process creates a output table that contains the following columns: - a character column labeled :SCI and containing the names of the scientific frames. Each name is repeated in nexp rows. - a character column labeled 'exptype' (i.e DARK,FF, etc) and containing the name of the associated exposures. - an integer column QUAL_'exptype' (i.e QUAL_DARK, QUAL_FF,etc) containing for each associated frame its quality flag.

4) The rule_table must contain the following columns: - FUNCTION -character column (max 256 char) - RANGE_1 -character column (max 256 char) - RANGE_2 -character column (max 256 char) - WEIGHT -R*4 column,

The column :FUNCTION shall contain mathematical expressions defining the different selection criteria in which the variables may be the columns of the OST. The expression may contain arithmetic and logical operators as well as mathematical functions. On top of the operators and functions supported by the SELECT/TABLE command, a new function REFVAL is available. REFVAL(:col) will return the value of the column 'col' for the corresponding scientific exposure. Column :RANGE_1 and :RANGE_2 shall contain the ranges defining the "first choice" and the "second choice" set of criteria, respectively. Column :WEIGHT shall contain the weighting factors for the different selection criteria.

For instance, a "first choice" selection criteria may in natural language be: " Find for each scientific image all calibration frames which have been observed within 3 days and for which the detector mean temperature does not differ by more than 1 degree " The "second choice" selection criteria may in that case be: " Find for each scientific image all calibration frames which have been observed within 15 days and for which the detector mean temperature does not differ by more than 10 degrees" These criteria would in the rule_table be expressed as:

```
FUNCTION RANGE_1 RANGE_2 WEIGHT
------------------------ ------- ------- ------
ABS(:DATE-REFVAL(:DATE)) <=3 <=15 0.7
ABS(:DETTEMP-REFVAL(:DETTEMP) <=1 <=10 0.3
------------------------ ------- ------- ------
```

**Examples:** `associates to each scientific frame listed in the table ost`
at most 3 DARK exposures fullfilling the criteria defined in the table dark_asso and stores the results in the table ntt_asso

# CLASSIFY/IMAGE

## CLASSIFY/IMAGE *stdred/do* MP 05-MAY-93

**Purpose:** classify images according to one or several classification rules.

**Syntax:** `CLASSIFY/IMAGE table descr outcol outchar (a)`

`CLASSIFY/IMAGE table classtable (b)`

| | |
|---|---|
| `table` | name of the Observation Summary Table |
| `descr` | name of the descriptor containing the classification rule |
| `classtable` | name of the table containing the classification parameters followed by the extension of the file (.tbl) |
| `outcol` | reference of the output column in the OST. Will be created with type C*8 if it doesn't exist. This column will contain the character string 'outchar' |
| `outchar` | character string for flagging the rows that satisfy the classification rule |

**See also:** CREATE/CRULE

**Note:** 1) A classification rule is a selection criterion for the SELECT/TABLE command. The classification process selects the rows satisfying the criteria and flags them with a given character string that is stored in the output column of the OST. The different rules are applied sequentially. Descriptors containing suitable classification rules may be created with the command CREATE/CRULE

2) The string outchar may contain the special character & which will be interpreted as a wild card replacement character, i.e if outchar is substring&no, the final character string will be built by finding the no-th occurence of the "*" character in the classification rule, reading the contents of the corresponding column and appending the result to substring. For instance, one has created the classification rule WFIB for flagging all exposures obtained in the blue arm of EMMI using the Wide-field imaging mode. One wants to flag these exposures with a character string containing the filter number that has been used. The WFIB descriptor may in natural language be: "Find all frames exposed in the blue arm using a filter and no grating" The translated selected criterion looks as follows: :FILTB_ID.EQ."*".AND.:FILTB_TYP.EQ."FILTER*".AND.:GRATB_ID.EQ."" The command CLASSIFY/IMAGE ntt WFIB :OPATH FB&1 will flag all files from the table ntt which satisfy this selection criterion and will store in the column OPATH the character string obtained by appending to the string FB the contents of column :FILTB_ID

3) When option b is chosen, a table classtable containing three character columns :DESCR, :OUTCOL and :OUTCHAR must be provided. Each row of this table should contain respectively the name of one classification rule descriptor, the label to the output column in the OST and the character string for flagging the rows satisfying the corresponding rule.

**Examples:** `CLASSIFY/IMAGE ost SCI :EXPTYPE SCI`
will flag all files from the table ost classified as scientific exposure with the character string SCI. The classification rule to be applied is taken from descriptor SCI of table ost.

## CREATE/CRULE *stdred/do* MP 05-MAY-93

**Purpose:** create an classification rule for a given Observation Summary Table

**Syntax:** `CREATE/CRULE table descr`

| | |
|---|---|
| `table` | input Observation Summary Table |
| `descr` | descriptor of input table that will contain the created rule. |

**See also:** CREATE/OST, EDIT/TABLE, CLASSIFY/IMAGE, SELECT/TABLE

**Note:** 1) The command creates and edits using EDIT/TABLE a temporary table containing two columns. The first one, labelled ":COLUMN" contains the labels of the columns of the input table. In the second one, labelled ":RULE" ,constraints to be applied to the values of these columns should be entered. Relational operators may be used (e.g., >, <, <=, != or =, default is =) as well as logical operators (&, —) Values or range of values have to be specified. ("*" is the wildcard character, " " ignores case and the ".." specifies a range of values). Constraints applied to more than one column will be ANDed.

2) These constraints will be converted into selection criteria suitable for the SELECT/TAB command and will be stored in the character descriptor descr (maximum length 256) of the table. The descriptor will be overwritten if it already exists. Furthermore, the command cannot store the selection criteria into the standard descriptors of the table (TBLENGTH, TBLOFFST, TBLCONTR, TSELTABL, TLABLxxx)

**Examples:** `CREATE/CRULE ntt DARK`
creates a rule for flagging BIAS exposures and stores the derived selection criteria into the descriptor BIAS of the table ntt.
The command will enter the table editor and the constraints on the values of the columns of the OST may be entered as follows:

```
COLUMN |RULE |
----------------------------------------
MJD | |
IDENT | = *BIAS*|*DARK* |
RA | |
DEC | |
EXPTIME | =0 |
----------------------------------------
```

These constraints will be translated into the following selection criteria:
:IDENT.EQ."* BIAS*.OR.:IDENT.EQ."* DARK*".AND.:EXPTIME.EQ.0 which is the format required by command SELECT/TABLE

# CREATE/OST

## CREATE/OST *stdred/do* MP 05-MAY-93

**Purpose:** create an Observation Summary Table

**Syntax:** `CREATE/OST file_specs [file_pref] intable outtable flag`

| | |
|---|---|
| `file_specs` | list of numbers of files to be processed (a) or file specification as used in the VMS '$ DIRECTORY' or UNIX '$ ls ' command (b) ( * matches zero or more characters, ? matches any single character, [] matches one of the enclosed characters) |
| `file_pref` | prefix of the files names. Used only with the option (a) of file_specs. The filenames will be built by appending to the prefix the file number and the file extension specified by "flag" |
| `intable` | name of the input table containing the list of the MIDAS descriptors to be mapped into the columns of the output table. |
| `outtable` | name of the Observation Summary Table |
| `flag` | 2-character flag: file format, append file format flag : F(its), the file extension .mt is assumed when the option a) is used. M(idas), the file extension is .bdf is assumed when the option a) is used. append flag: A(append to an existing OST) C(reate a new OST) Default flags are FC. |

**See also:** CREATE/TABLE

**Note:** 1) The Input Table must contain the followings columns:

```
DESCR_INAME (Character Column):  contains the list of MIDAS
descriptors to be mapped
into the columns of the OST.
IPOS (Integer Column) :  contains for each descriptor
the position of the element
to be read.
DESCR_ONAME (Character Column):  contains for each descriptor
the label of the column of the
OST in which will be stored its
values
OTYPE (Character Column):  contains for each descriptor
the type of the column of the
OST in which will be stored its
values
I (integer), R(eal), D(double
precision), C*n(character string)
```

2) The command maps each of the MIDAS descriptors from the column DESCR_INAME of the Input Table into one column of the Output Table. The values of these descriptors are read from each of the files of file_spec and are stored into one row of the Output Table. If a descriptor contains more than one element, the one at the position defined in the column :IPOS is taken.

3) The way FITS keywords are translated into MIDAS descriptors is decribed in the Chapter 7 of Volume A.

**Examples:** `CREATE/OST ntt3*.mt ? ntt_descr ntt_ost`
Process all the FITS files whose names match the pattern ntt3*.mt Read for each of them the values of the descriptors listed in column :descr_iname of table ntt_descr and store them in table ntt_ost

`CREATE/OST 3-5 ntt ntt_descr ntt_ost MA`
Process the MIDAS images ntt0003.bdf, ntt0004.bdf, ntt0005.bdf, Read for each of them the values of the descriptors listed in column :descr_iname of the table ntt_descr and append them in the existing table ntt_ost

---

## GROUP/ROW *stdred/do* MP 05-MAY-93

**Purpose:** group the rows of a table that have the same value in a given column and assign to them a group number (integer) that is stored into an output column

**Syntax:** `GROUP/ROW table incol outcol`

| | |
|---|---|
| `table` | table name |
| `incol` | reference of the input column |
| `outcol` | output column (will be created if it doesn't exist) |

**See also:**

**Note:** The input column may be of any type. If it contains floating point numbers (R*4 or R*8), the row values are truncated before being grouped.

**Examples:** `GROUP/ROW ntt :EXPTYPE :EXPTYPE_G`
group the rows of the table ntt that have the same value in the column EXPTYPE and store the group number into the column EXPTYPE_G

# Appendix I

# Standard Reduction - irac2

# Standard Reduction Commands

# Context: irac2

# ACUTS/IRAC2

## ACUTS/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Display an image with cuts mean-3*sig and mean+upper*sig

**Syntax:** `ACUTS/IRAC2 [image] [load] [plot] [upper]`

| | |
|---|---|
| `image` | name of input image. Default is the name of the image in the keyword IDIMEMC |
| `load` | flag for image loading. Default is "Y" |
| `plot` | flag for plotting image histogram. Default is "P" for plotting "N" is for not plotting |
| `upper` | upper cutoff in units of the standard deviation. Default = 8 |

**Examples:** `ACUTS/IRAC2 ir_frame ? P`
Load the image ir_frame and plot the image histogram

## CMASK/IRAC2 *stdred/irac2* 09-JUL-1992 GF

**Purpose:** Create a mask of bad pixels using a flatfield.

**Syntax:** `CMASK/IRAC2 ffield clnffield lthrhold,hthrhold [dispflag]`

| | |
|---|---|
| `ffield` | name of image file which is used to create the mask of bad pixels. |
| `clnffield` | name of file which contains the clean flatfield having all bad pixels replaced by neighbouring good pixels or set their pixelvalue to hthrshold depending on dispflag. |
| `lthrshold` | minimum pixel value for good pixels; all pixels having a pixel value smaller than lowthreshold are considered as bad pixels |
| `hthrshold` | maximum pixel value for good pixels; all pixels having a pixel value larger than highthreshold are considered as bad pixels |
| `dispflag` | R or H : R replaces all bad pixels by neighbouring good pixels. H sets the pixelvalue of bad pixels to highthreshold. Default is H. |

**Note:** The pixelmask is stored in file GOODPX.bdf which is used by the MIDAS command MASK/IRAC2. Good pixels in file GOODPX.bdf are marked by 1 and bad pixels are marked by 0.

**Examples:** `CMASK/IRAC2 flatfield cleanflatfield 3000,7000 R`
Use frame flatfield to create pixelmask. All pixels of flatfield which have an intensity bellow 3000 or above 7000 are considered to be bad. The bad pixels of the frame flatfield.bdf are replaced by neighbouring good pixels and the resulting image is stored in the frame cleanflatfield.bdf which is displayed using the cut levels of frame flatfield.bdf.

DCOMB/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Sky subtract and combine dithered images.

**Syntax:** `DCOMB/IRAC2 [select] [seqname] [accsky] [align] output [trim] [tag]`

| | |
|---|---|
| `select` | selection string which selects files in the OST table. For example @21..24,29 will select files in row numbers 21 to 24 inclusive and file in row number 29. Default = null. |
| `seqname` | files with this sequence number will be selected. Default is null. |
| `accsky` | Flag for accurate sky subtraction. Default is N. |
| `align` | Flag for aligning object frames. Options are:<br>Y for interactive alignment;<br>A for automatic alignment;<br>B for blind alignment and<br>N for no alignment.<br>Default is Y. |
| `output` | name of output image. |
| `trim` | flag for trimming image to largest common area. Default is Y. |
| `tag` | flag for tagging all frames marked as skies or tagging only those with the word "sky" in the identifier. Default is Y. |

**Note:** Out of the input parameters select and seqname only one should be set.
If the flag for accurate sky subtraction is set to Y, the user is prompted to enter a number which determines which pixels are replaced before image smoothing. Use a low value if the sky is stable and use a high value if the sky is varying rapidly. However, the higher the value, the more inaccurate photometry becomes.
The first few characters of the output name must be different than the prefix of the raw data files.
If the flag for tagging frames as skies is set to N, the identifier of the frames to be used as skies must be marked as sky.
The name of the output image is recorded in the MIDAS table reductions.
For IRAC2 data, option B is not recommended.

**See also:** RCOMB/IRAC2

**Examples:** `DCOMB/IRAC2 ? 342 N Y hd1234 N Y`
Using the images from sequence 342, combine all the images to make a sky, subtract this sky image from all images, and combine all these images into the final image with the user selecting objects for image registration. The final image is called hd1234 and is trimmed to the largest common area of the input images.

# FFIELD/IRAC

## FFIELD/IRAC *stdred/irac2* 5-Oct-1995 DLC

**Purpose:** Flat fields a data frame by dividing it by a previously generated flat field

**Syntax:** `FFIELD/IRAC obj_frame ff_frame out_frame`

| | |
|---|---|
| `obj_frame` | name of the object frame |
| `ff_frame` | name of flat field frame |
| `out_frame` | name of output frame |

**Note:** This command divides the input frame by the flat field frame and sends the result to the output frame. The flat field frame should have previously been calculated using make_iracflat or by using COMPUTE/IMAGE, and should be normalised to 1.

**Examples:** `FFIELD/IRAC im0010 kflat imf0010`
Use the flat field kflat to flat field the frame im0010. The output frame is imf0010

## FOCUS/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Used to determine the best focus from a focus sequence.

**Syntax:** `FOCUS/IRAC2 seqnum [focout] [create]`

| | |
|---|---|
| `seqnum` | sequence number |
| `focout` | output name of the reduced focus sequence. Default name = focus |
| `create` | flag to decide if the file listed in the variable focout is created. The alternative is that a previously created file, as listed in the variable focout, is used. Default = Y. |

**Note:** The command requires a focus sequence and the function file "focfunction.fit".

**Examples:** `FOCUS/IRAC2 5`

## LAST/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Gives very brief information on the most recent exposures

**Syntax:** `LAST/IRAC2 [num]`

| | |
|---|---|
| `num` | information on the last num exposures are displayed. Default is 10 |

**Note:** The programme lists the contents of the image catalog "data.cat"

**See also:** CREATE/ICAT

**Examples:** `LAST/IRAC2 5`
List the information about the last 5 exposures

## MASK/IRAC2 *stdred/irac2* 09-JUL-1992 GFI

**Purpose:** Replace each bad pixel by closest good pixel.

**Syntax:** `MASK/IRAC2 inframe outframe`

| | |
|---|---|
| `inframe` | name of image file which is to be cosmetically treated by replacing bad pixels by neighbouring good pixels. |
| `outframe` | name of resulting file which contains the clean image having all bad pixels replaced. |

**Note:** The pixelmask is stored in file GOODPX.bdf which has to be created by the MIDAS command CMASK/IRAC2 in the current directory. Good pixels are marked by 1 and bad pixels are marked by 0 in file GOODPX.bdf.

**Examples:** `MASK/IRAC2 spiral spiralclean`
The bad pixels of the frame spiral.bdf are replaced by neighbouring good pixels using the pixel mask GOODPX.bdf which has been created by the midas command CMASK/IRAC2. The clean image is stored in the frame spiralclean.bdf and displayed using the cut levels of frame spiral.bdf.

## MKFLAT/IRAC *stdred/irac2* 6-Oct-1995 DLC

**Purpose:** Subtracts lamp off from lamp on and normalises result to make a flat field

**Syntax:** `MKFLAT/IRAC lamp_on lamp_off flat_field`

| | |
|---|---|
| `lamp_on` | name of frame with flat field light on |
| `lamp_off` | name of frame with flat field light off |
| `flat_field` | name for output flat field frame |

**Note:** This command subtracts a lamp off frame from a lamp off frame and then normalises the resulting frame to 1 to generate a flat field. The user should check that the filters and integration times for the input images are the same, otherwise the output will not be scientifically useful.

**Examples:** `MKFLAT/IRAC klampon klampoff kflat`
Calculate the flat field image kflat by subtracting the lamp off frame klampoff from the lamp on frame klampon and then normalising the resulting frame to a mean of 1.

# OBSLIST/IRAC2

## OBSLIST/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Lists a subsection of the IRAC2B OST (Observation Summary Table)

**Syntax:** `OBSLIST/IRAC2 [start] [end]`

| | |
|---|---|
| `start` | starting OST column number |
| `end` | ending OST column number |

**Note:** This command uses the OST column number and not the number in the file name. These numbers usually agree, but not always. If both parameters, start and end, are left blank, the last ten entries are displayed. If the parameter end is left blank the ten entries from start onwards are displayed. The format of the output is dictated by the file obslst.fmt. The name of OST is irac2b_ost.tbl.

**See also:** OBSREP/IRAC2, CREATE/OST

**Examples:** `OBSLIST/IRAC2 4 7`
List column 4 to 7 of the observation summary table irac2b_ost.tbl

## OBSREP/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Print out a subsection of the IRAC2B OST (Observation Summary Table)

**Syntax:** `OBSREP/IRAC2 start end`

| | |
|---|---|
| `start` | starting OST column number |
| `end` | ending OST column number |

**Note:** This command uses the OST column number and not the number in the file name. These numbers usually agree, but not always.

The format of the output is dictated by the file obsrep.fmt. The name of OST is irac2b_ost.tbl.

**See also:** OBSLIST/IRAC2, CREATE/OST

**Examples:** `OBSREP/IRAC2 4 7`
Print column 4 to 7 of the observation summary table irac2b_ost.tbl

## QL/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Subtracts one IRAC2 image from another and divides by the detector integration.

**Syntax:** `QL/IRAC2 image1 image2 [outimage]`

| | |
|---|---|
| `image1` | file number of first image |
| `image2` | file number of second image |
| `outimage` | output image name. Default name is "out" |

**Note:** This programme uses the number in the file name. It requires the header of the filenames to be "soft-wired" into the programme.

**Examples:** `QL/IRAC2 3 5 3min5`
Subtract number 5 from number 3 and divide by the detector integration. The output image will be "3min5".

## RCOMB/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Combine frames created with the command RCOMB/IRAC2

**Syntax:** `RCOMB/IRAC2 select [align] output`

| | |
|---|---|
| `select` | selection string which selects files in the table reductions. For example, @21..24,29 will select files in row numbers 21 to 24 inclusive and file in row number 29. |
| `align` | Flag for aligning object frames Options are:<br>Y for interactive alignment;<br>N for no alignment.<br>Default = Y. |
| `output` | name of output image. |

**Note:** The name of the output image is recorded in the MIDAS table reductions.

**See also:** DCOMB/IRAC2

**Examples:** `Combine the frames 10 to 14 and 16 into the output frame redcomb`
without alignment.

## SEEING/IRAC2 *stdred/irac2* 1-Nov-1995 CEL

**Purpose:** Determine the seeing, defined as the FWHM of stellar images, of IRAC2 images.

**Syntax:** `SEEING/IMAGE`

**Examples:** `SEEING/IMAGE`

# SSUB/IRAC

## SSUB/IRAC *stdred/irac2* 6-Oct-1995 DLC

**Purpose:** Subtracts a sky frame from an object frame.

**Syntax:** `SSUB/IRAC obj_frame sky_frame out_frame`

| | |
|---|---|
| `obj_frame` | name of the object frame |
| `sky_frame` | name of sky frame |
| `out_frame` | name of output frame |

**Note:** This command subtracts a previously generated sky frame from an object frame to generated a sky-subtracted image of the object.

**Examples:** `SSUB/IRAC im0010 im0011 ims0010`
Subtract the sky frame im0011 from the object frame im0010 to generate the sky-subtracted object frame ims0010.

# Appendix J

# Standard Reduction - irspec

# Standard Reduction Commands

# Context: irspec

# BADPIX/IRSPEC

## BADPIX/IRSPEC *stdred/irspec* 21-SEP-1992 EO

**Purpose:** The IRSPEC array has a fixed pattern of bad pixels. This command cleans the image by substituting the values of the bad pixels with the average of the neighbouring good pixels. The fixed-pattern of bad pixels can be defined using DEFINE/IRSPEC and updated using SET/IRSPEC badpix=.....

**Syntax:** `BADPIX/IRSPEC in out [load=load_opt.] [clean=clean_opt.]`

`[debug=debug_opt.]`

| | |
|---|---|
| `in` | input image; untouched on exit. Must have the original size with starts and steps all equal to 1. It could also be an operation between images (see last example). |
| `out` | name of the output image. |
| | Additional parameters and options: |
| `load` | 0,1 default=0. If =0 (the default) the command works silently. if =1 it will display the image before and after the correction |
| `clean` | a,x,y,b default=a. If = a (the default and recommended value) it decides automatically, for each bad pixel, whether to use the average of the good pixels along x, y or b (x+y). The user can force to use any of the x,y,b options by setting dir=x, dir=y or dir=b. |
| `debug` | 0,1 default=0. Normally = 0, to be set to 1 only for reporting errors in case the command does not work properly in the automatic mode. |

**Note:** BADPIX works only with fairly isolated bad pixels and cannot handle cases when the bad pixels are clustered, in which case it will give an error message. In such a situation use FILTER/MEDIAN and similar commands.

**Examples:**

```
BADP/IRS a0015 out1

BADP/IRS myframe octopus c=x l=1
```
(note that l=.. c=... are enough for clean=.. and load=...)

```
BADP/IRS (cc0012+cc0015-cc0013-cc0014)/2.  out2 l=1
```

## CALIBRATE/IRSPEC *stdred/irspec* 22-SEP-1992 EO

**Purpose:** The wavelength dispersion on the IRSPEC array is linear within a small - and totally negligible - fraction of the pixel size. Hence, wavelength calibrating simply means modifying the x-start and x-step values (descriptor) of the image. Another advantage of IRSPEC is that one can very precisely compute (analytically) the pixel size - in wavelength - once the central wavelength of the frame is known. A quite precise - usually within 1 pixel - estimate of this quantity is available on-line at the instrument ("mechanical" calibration) and is stored in the original files in the form of descriptors. You can directly use this information and determine the "mechanical" wavelength calibration as in the first example. This command allows you also to determine more precisely the central wavelength of your frame - and hence to obtain a very accurate wavelength calibration - if you have a frame containing lines with known wavelengths; up to 2.3 microns the OH lines in the sky frames are a very convenient calibrator (Oliva & Origlia, 1992, A&A 254, 466). See the 2-nd example and/or use TUTORIAL/CALIBRATE. Finally, this command can be used to apply a precise calibration to a given image (see third example).

**Syntax:** `CALIBRATE/IRSPEC ima [ref=reference_ima] [mode=calibration_mode]`

`[units=wavelength_units]`

| | |
|---|---|
| `ima` | Frame to wavelength calibrate, it may have any size in Y (could e.g. be 1D) but must have the original size in X. "Untouched" on exit (see last note below) If mode=d (definition of precise calibration), "ima" is the sky/lamp frame containing lines with known wavelengths and must have the original X,Y sizes. Untouched on exit. |
| `ref` | Optional reference image which contains precise calibration parameters, determined using CALIB/IRSP with mode=d |
| `mode` | If set to d (define) it interactively defines the best wavelength calibration parameters using the lines contained in the frame "ima" (see also TUTORIAL/CALIB). Use mode=d (or just m=d) for calibration using sky lines (you need a flat and a dark for this). For lamp lines frames it may be more convenient to work on the frame as it is, in which case use m=d1 (sorry for the dirty trick...). |
| `units` | Wavelengths are given in microns. You can use units=a (or just u=a) if you prefer to work in Angstroms. |

**Note:** It is most convenient (and clearer) to apply the wavelength calibration only to "rectified frames" - i.e. with straight spectral lines (see RECTIFY/IRSPEC).
The wavelength calibrated frame is identical to the original but has different start and step in X. Therefore, if you want to go back to the uncalibrated image it's enough using WRITE/DESC image_name start 1 WRITE/DESC image_name step 1

**Examples:** `CALIB/IRSP test4`
apply mechanical calibration to frame test4

`CALIB/IRSP sky3 mode=d`
define and store in "sky3" precise calibration parameters from the sky/lamp lines contained in the same frame (see also TUTORIAL/CALIB)

`CALIB/IRSP test4 ref=sky3`
apply precise parameters (defined above) to frame "test4"

# DEFINE/IRSPEC

## DEFINE/IRSPEC *stdred/irspec* 21-SEP-1992 EO

**Purpose:** The IRSPEC array has a fixed pattern of bad pixels which can be corrected using BADPIX/IRSPEC once the positions of the bad pixels are stored in a table. This command is used to define the bad pixels and store their positions into a given table, which can be then selected using SET/IRSPEC badpix=..... Note that a "standard" table of bad pixels (named – irsbadpix – ) is automatically created after the SET/CONT IRSPEC command and is used by default.

**Syntax:** `DEFINE/IRSPEC in_ima out_tab [mode=selection_mode] [t=threshold]`

`[n=#of_subframes] [load=load_option]`

`in_ima` input image (usually a dark frame). Must have the original size with starts and steps all equal to 1. Untouched on exit.

`out_tab` name of the output table containing the coordinates of the bad pixels.

Additional parameters and options:

`selection_mode`

r,a. If = r (relative, the default) the command select the bad pixels as those with abs(value-average) > threshold*sigma where "threshold" is the next parameter while "average" and "sigma" are the corrected mean and standard deviation of a subsection of the image whose size is defined by the second next parameter. If = a (absolute) the bad pixels will be those with value<threshold(1) or value>threshold(2) where "threshold(1:2)" are the next parameter.

`threshold` number[,number]. If mode=r a single number (default=15) if mode=a two numbers separated by a comma

`#of_subframes` integer_number. Used only when mode=r, in which case the "average" and "sigma" defined above are separately computed in NxN subframes

`load_option` 0 or 1. If = 1 (defaults) it displaies the positions of the bad pixels found. Use l=0 to force the command to work silently.

**Note:** You will probably find it easier to understand the role of the additional parameters by trying to use them rather than reading the above descriptions.

**Examples:**

```
DEF/IRS dark01 badpix1

DEF/IRS dark01 badpix2 mode=a t=9 n=4

DEF/IRS dark01 badpix3 mode=r t=30,900
```

## FLAT/IRSPEC *stdred/irspec* 21-SEP-1992 EO

**Purpose:** To create a normalized flat from an halogen frame. It can be stored and made available for use typing SET/IRSPEC flat=.... See also SKYSUB/IRSPEC, TUTORIAL/SKYSUB and CALIBRATE/IRSPEC

**Syntax:** `FLAT/IRSPEC in_flat in_dark out [load=load_opt.]`

`[thres=threshold] [vignet=vignetted_value]`

| | |
|---|---|
| `in_flat` | input flat image. Must have the original size with starts and steps all equal to 1. Untouched on exit. |
| `in_dark` | optional name of a dark frame (with same integration time as the flat); set it to – 0 – if no dark is needed (see first example). Untouched on exit. |
| `out` | name of the output, normalized flat image. This is automatically corrected by the bad pixels (see BADPIX/IRSPEC). |
| | Additional parameters and options: |
| `load` | 0 or 1 default=1. If =1 (default) it will display the images during the various phases of the process. if =0 the command works silently. |
| `threshold` | real positive number, default=0.5. All the rows in the input image with average values lower than threshold*aver_cen ("aver_cen" being the average value of the central 6 rows) are considered as vignetted and will be forced to "vignetted_value" (the next parameter). |
| `vignetted_value` | real number, default=100. Value assigned to all the pixels in vignetted rows (defined through the previous parameter) in the output image. |

**Note:** The normalization procedure just consists of dividing the original frame, cleaned of the bad pixels, by the average value of the central 6 rows (this value is also stored in the identifier starting from the 20-th character). Hence, the normalized output image contains also the detector response at low spatial frequencies. Use FIT/FLAT or similar commands if you need a "more normalized" flat. The vignetted rows are set to large values in order to make them vanishing after division by the flat.
For applications up to 2.5 microns the input, flat image is usually a measurement of the halogen lamp with counts level as close as possible - and well within a factor of 2 - to those in the astronomical frames. At longer wavelengths the sky is so bright that you may use the sky itself as flat source.

**Examples:**

```
FLAT/IRS halo0001 0 flat1

FLAT/IRS halo0002 dark1 flat2 v=1000 l=1

FLAT/IRS halo0003 0 flat3 t=0.3 v=50
```

(note that l=.. t=.. v=.. are enough for load=.. thres=... vignet=..)

# FLUX/IRSPEC

## FLUX/IRSPEC *stdred/irspec* 24-SEP-1992 EO

**Purpose:** To flux calibrate a spectrum which could either be a 2D long-slit image or a 1D spectrum. You must have previously created a "response frame" using RESPONSE/IRSPEC.

**Syntax:** `FLUX/IRSPEC in_ima response_ima out_ima [smooth=s1,s2]`

`[shift=sh] [norm=normalize_option]`

`in_ima` input image. Could be either 2D (a long-slit frame) or 1D (a spectrum) and must be wavelength calibrated (see CALIBRATE/IRSPEC). It must not be normalized unless you use the normalize_option (see below). In case of 2D long-slit spectrum the command also rectifies the image if you have forgotten it. N.B. the input frame is untouched on exit.

`response_ima` input (1D) response image previously created with RESPONSE/IRSPEC. It must obviously refer to standard star data taken at the same grating position.

`out_ima` name of the output, flux calibrated spectrum. It will have the same size and limits as the input frame, i.e. it will be a 2D flux calibrated spectrum in case the input was a long-slit frame.

Additional parameters and options:

`smooth` If you want to smooth along X (the dispersion direction) the object image and/or the response frame use smooth=s1,s2 (or just sm=s1,s2) where s1,s2 are the amount - in pixels - by which you want to smooth the object and the response frame, respectively. The smoothing may help in the reduction of spectra at wavelengths where the atmospheric transmission is very bad. Be however careful not to abuse it because you may easily "invent" good data from bad spectra.

`shift` This parameter allows you to shift the star (response) spectrum by a given fraction of a pixel before flux calibrating; it may be of some use for data at wavelengths with very bad atmospheric transmission. Use e.g. shift=0.2 (or just sh=0.2) to apply a shift of 0.2 pixels. The default value is obviously sh=0.

`normalize_option`

0,1; default=1. The command automatically takes into account, and corrects for, the on-chip integration time of the object frame. Use norm=0 (or just n=0) to force the command to skip the automatic, normalization procedure.

`rectify_option`

0,1 default=1. In case the input object frame is 2D, the command checks if the image was rectified - using RECTIFY/IRSPEC - and performs this operation in case this was not previously done. Use rect=0 (or just r=0) to skip this control.

**Note:** The command checks if the response frame is a "proper" response frame by looking at a descriptor in the response image. If you get strange messages it may be that your response frame is the result of of a series of non standard operations in which the descriptor got lost. Use COPY/DD original_response_ima *,3 your_image to recover it (original_response_ima is the output of the RESPONSE/IRSPEC command). A similar argument applies to the normalization and to the "rectify-control" (see also the help of RESP/IRSPEC).

**Examples:**

```
FLUX/IRS objh8 resph8 fluxh8

FLUX/IRS objk01 respk01 fluxk01 smooth=1,1 shift=-0.15
```

## MERGE/IRSPEC *stdred/irspec* 22-SEP-1992 EO

**Purpose:** To merge (connect) overlapping 1D spectra (images) into a table. It also allows optimizing the connection on the overlapping parts and excluding a given number of pixels at both edges of each spectrum.

**Syntax:** `MERGE/IRSPEC prefix_ima i1,i2[,i3] out_table`

`[excl=#pixels_excluded] [corr=correct_option]`

`[ref=#reference_image] [plot=plot_option]`

`[format=i_format]`

`prefix_ima` The 1D spectra to be connected must be named with a prefix and a number, e.g. spec0001 spec0002 ... prefix_ima is simply the image prefix ( - spec - in the above example).

`i1,i2[,i3]` The command merges the frames "prefix"i1 to "prefix"i2 with step i3 (default=1), see the examples. The spectra must be ordered in wavelength from "i1" to "i2" (step "i3") and overlapping. However, randomly distributed spectra may also be handled if no correction is applied (see correction_ option below).

`out_table` Name of the output table containing the merged spectrum. It can be transformed into a 1D image using CONVER/TABLE.

Additional parameters and options.

`#pixels_excluded`
integer number, larger or equal 0, default=0. Can be used to exclude a fixed number o pixels at both edges of each spectrum.

`correct_option`
0,1 default=1. If equal to 1 (default) the command forces the spectra to properly connect on the overlapping parts. If equal to 0 it just merge together the spectra into the table without any correction.

`#reference_image`
integer number, default=0. In the process of forcing the connection of the overlapping regions one of the 1D images (the "reference image") is left untouched. If ref=0 (the default value) the reference image is the central 1D frame of the i1,i2[,i3] list. Use ref=image_number to specify a different reference image, e.g. if your images are spec0010 spec0011 ... ref=11 will select spec0011 as reference.

`plot_option` 0,1 default=1. If equal to 1 (default) it plots the spectra during all the phases of the procedure. Be careful to use SET/GRAP xaxis=... yaxis=... to properly include all spectra in the plot. If equal to 0 it works silently.

`i_format` i"n" , default i4. To be used in case the names of your images contain numbers shorter/longer than 4 digits, e.g. for spec01 spec02 ... use format=i2 (or just f=i2).

**Note:** The connection between the spectra is optimized by applying a multiplicative factor to the spectra (N.B. the input spectra are left untouched).This implies that this command works properly only with spectra with strong and well defined continua while it may produce totally unreliable results with spectra with faint continua affected by variable offsets.

**Examples:** `MERGE/IRS a 12,24,2 outa`
merge spectra a0012,a0014,a0016..a0024 into table outa.

# MERGE/IRSPEC

`MERGE/IRS a 12,24,2 outa e=4`
as above but excluding 4 pixels at both edges of each spectrum.

`MERGE/IRSP b 11,18 outb p=0 f=i2`
merge spectra b11,b12,b13...b18 into outb, works silently.

`MERGE/IRSP c 50,5,-5 outc c=0`
merge c0050,c0045,...c0005 into outc without applying corrections.

RECTIFY/IRSPEC *stdred/irspec* 22-SEP-1992 EO

**Purpose:** The slit images at the various wavelengths ("spectral lines") are tilted as a consequence of the off-axis mount of the grating, and the angle by which they are tilted varies with the position of the grating, i.e. with the wavelength. This command is intended to correct for this effect and hence "rectify" the spectral lines. The tilt angle is computed analytically from the instrumental calibration parameters (on line central wavelength etc.) which are stored as descriptors in the original frames. The user can also choose the value manually.

**Syntax:** `RECTIFY/IRSPEC in out [load=load_opt.] [tilt=tilt_value]`

`[ref=reference_row]`

`in` input frame. Must have the original size with starts and steps all equal to 1. Untouched on exit.

`out` name of the output, rectified image.

Additional parameters and options:

`load` 0,1 default=0. If =0 (the default) the command works silently. If =1 it will display the image before and after tilting.

`tilt` real number, default=0 If =0 (default) the command computes automatically the tilt angle. Use any other value to force the program to apply a given tilt angle.

`reference_row` integer number, default=0. Position of the "pivotal" (or "fulcrum") row which is left untouched by the tilt procedure. If =0 (the recommended default) it takes the middle row of the frame.
You can also use any positive number (not larger than the Y-size of the array) but, if you do, you may get into troubles with the wavelength calibration (see also TUTORIAL/CALIBRATE).

**Note:** If you use the automatic (default) choice of the tilt angle and get strange error messages it may be that your input frame (e.g. the result of a long series of operations) has lost the descriptors which are needed by the program. Try then
COPY/DD orig *,3 frame
where "frame" is your image and "orig" is (one of) the original frames taken at the same grating position.
The X-sizes of the input and output frame are the same. This means that, after applying RECTIFY/IRSPEC, you lose information about 2 opposite corners of your image and "invent" data in the other two edges.
To be precise, the spectral lines are section of non-parallel parabolae. However, one can verify that approximating them as parallel lines introduces errors well below 1 pixel over the entire array. The analytical value of the tilt angle should also be accurate within a fraction of a pixel

**Examples:**
```
RECT/IRS a0012 obj1

RECT/IRS a0012 obj2 t=-0.12 l=1
```
(note that tilt=.. etc. can be abbreviated with t=... etc.)
```
RECT/IRS a0012 obj3 r=33 l=1
```

# RESPONSE/IRSPEC

## RESPONSE/IRSPEC *stdred/irspec* 24-SEP-1992 EO

**Purpose:** To a create a (1D) response frame which contains the conversion from counts/sec to flux units for the measurements taken at a given central wavelength (grating position). It requires a "sky-subtracted" and wavelength calibrated 2D frame of a "standard" star plus a flux table - previously created with STANDARD/IRSPEC - containing the intrinsic star fluxes. It automatically takes into account, and correct for, the on chip integration time so that the output response frame will contain the counts/sec given by a given flux unit.

**Syntax:** `RESPONSE/IRSPEC in_ima flux_table out_response_ima`

`[yrows=y1,y2,y3,y4] [obs=observation_mode]`

`[norm=normalize_option] [rect=rectify_option]`

`in_ima` Input 2D frame containing the observed spectrum(a) of the reference star. It must be cleaned, flat fielded "sky-subtracted" (e.g. the result of SKY-SUB/IRSPEC, see also the observing_mode below) and wavelength calibrated (see CALIBRATE/IRSPEC). It must not be normalized unless you use the normalize_option (see below). The command also rectify the image if you have forgotten it, unless you used rect=0 (see below). N.B. the input frame is untouched on exit.

`flux_table` Input table - created using STANDARD/IRSPEC - containing wavelengths and star fluxes (columns :wl :flux). In case you are only interested in equivalent widths or normalized spectra and you do not want to be bothered with flux calibration you can set this parameter to - 0 - (see last example). Note that in such a case the average value of your response frame will be forced to unity.

`out_response_ima`
Output 1D image containing the instrumental response (counts/sec per unit flux) or just the 1D star spectrum, if you set the flux_table to 0. It will be used in FLUX/IRSPEC.

Additional parameters and options

`yrows` Y-positions (rows) which has to be summed to obtain the 1D stellar spectrum. You need 4 numbers because the program expects to have 2 star spectra - one positive and one negative - in the 2D frame. If you do not specify anything you will be asked to define the four positions with the display cursor. In case your star observations were of the type "object-sky" (see parameter obs=... below) you still need 4 row (cursor) positions, 2 to define the star and 2 to define the sky.

`observing_mode`
"ab" or "os", default=ab. The most usual and recommended observing procedure is to take "object" and "sky" frames with reference to the star set at two different positions along the slit. In this case you will have 2 spectra, one positive and one negative, in your frame. For these kind of frames use obs=ab (default). If you have just one spectrum in your 2D image you must use obs=os (o=o is enough) otherwise the response values will be a factor of 2 too low.

`normalize_option`
0,1, default=1. The command automatically takes into account, and corrects for, the on-chip integration time so that the output response frame will contain the counts/sec given by a given flux unit (in short it "normalizes" the observed counts). Use norm=0 (or just n=0) to force the comma to skip the automatic, normalization procedure.

`rectify_option`

0,1, default=1. The command automatically checks if the input 2D image was rectified - using RECTIFY/IRSPEC - and performs this operation in case this was not previously done. Use rect=0 (or just r=0) to skip this control.

**Note:** To normalize the frame to counts/sec the command reads the on-chip integration time which is stored in the original frames as a descriptor. If you get strange error messages (e.g. descriptor not present) it probably means that your input image is the result of a series of non standard operations in which the descriptor got lost. Use
COPY/DD original_ima *,3 your_image
to recover it. A similar argument applies to the "rectify-control" which may not recognize that the image was already rectified if the proper descriptor got lost into non standard operations after the the RECTIFY/IRSPEC command. Use COPY/DD ... *,3 .... in this case too.
The Y-positions of the rows that you have defined with the cursor are stored in the output frame in the descriptor IRS_YROWS, to look at them use
READ/DESC frame_name IRS_YROWS
In case the reference star has obvious, intrinsic features you should correct them in the response file. There is no - or at least I could not find - a standard approach to it. You may fit gaussian to the absorption line but this works easily only for strong and isolated features in regions of good atmospheric transmission, in which case modifying the data manually using MOD/GCURS may give similarly good results. A convenient approach could be that of having two response files from stars which you can assume not to have lines in common (e.g. a very hot and a fairly cool type) in which case by dividing the two frames you should be able to recognize and correct the star features as they appear in absorption or in emission depending to which of the two stars they belong. But this means that you should have already planned your observations in advance and, indeed, "lost" quite a bit of time just looking at the stars...

**Examples:** `RESP/IRSP bgdiff hr4138 bgresp`
Create response frame "bgresp" from 2D image "bgdiff" and flux table "hr4138" using the cursor to specify the Y-positions of the star spectra

`RESP/IRSP bgdiff hr4138 bgresp y=23,28,44,49`
Same as above but with the explicit specification of the Y-positions (note that y=... is enough).

`RESP/IRSP bgdiff hr4138 bgresp y=23,28,44,49 obs=o`
As above but with the original frame containing just one star spectrum (i.e. with the star out of the slit in the original sky frames).

`RESP/IRSP pbstd 0 pbrnoflux y=19,25,48,53`
Create a "non-flux-calibrated" response frame called "pbrnoflux" which just contains the observed star spectrum (in counts/sec).

# SKYSUB/IRSPEC

## SKYSUB/IRSPEC *stdred/irspec* 23-SEP-1992 EO

**Purpose:** In most of the infrared the sky emission contains bright emission lines, OH transitions up to 2.3 microns and molecular bands (CO_2, CH_4 and others) at longer wavelengths. The intensity of the OH transitions varies with time, which implies that you are never sure that your "obj-sky" frame will be free of residual sky lines. Besides, it sometimes occurs that the grating moves by tiny amounts and produces very small shifts - usually within a few hundreds of a pixel - of the lines. Even though small, these shifts deteriorate the quality of the final image where the detector noise is often well below the 1 percent of the sky line intensity. This command is intended to correct the "obj-sky" frame both for the time variation of the sky line and for the shift. The way it does it is to multiply the sky frame by a given quantity ("factor") and to X-shift the object frame by a given amount ("shift"). These parameters can be chosen automatically by the program, interactively - with the graphic cursor - or just entered manually; all depending on the value of "factor" you enter (use TUTORIAL/SKYSUB for a thorough explanation).

**Syntax:** `SKYSUB/IRSPEC ima_obj ima_sky out factor[,shift[,deltax,deltay]]`

`[sky=sky_table] [force=force_sky_to_zero]`

`[cuts=cuts_values] [debug=debug_option]`

`ima_obj` Input object frame, must have the original X,Y sizes; untouched on exit. It could also be an arithmetic expression between images (last example).

`ima_sky` Input sky frame, must have the original X,Y sizes; untouched on exit. It could also be an arithmetic expression between images (last example).

`out` Output frame, which is "obj-sky" cleaned by bad pixels, flat-fielded and corrected for sky lines according to the user's choice.

`factor[,shift[,deltax,deltay]]`
factor controls whether the values of "factor" and "shift" are defined automatically, manually or interactively. Please use TUTORIAL/SKYSUB to get a full description of all the possibilities and to to get examples of the uses of the 4 parameters.

`sky_table` optional table containing the regions of the image which do not contain any object emission (use TUTORIAL/SKYSUB for more information).

`force_sky_to_zero`
Normally is 0, if set to 1 subtracts from the final image the average value of the pixels in the sky regions (use TUTORIAL/SKYSUB for more information).

`cuts` cuts to be used in loading the images (as in LOAD/IMA), use cuts=n to force the command to work silently (this is not always possible...)

`debug` normally is 0, to be set to 1 only for reporting errors in case the command does not work properly in the automatic mode.

**Note:** The output frame is also flat-field and cleaned of the bad pixels, this means that you may find it convenient to use SKYSUB even for making a straight image subtraction (see second example).
A flat and a dark must be previously stored using SET/IRSPEC, the dark being necessary for the fact that when you multiply the sky by a number different the unity your final image will have a residual "dark" which has to be corrected for. Obviously, when working beyond 2.3 micron the dark has little sense because of the strong thermal background which dominates over the internal dark.
The effect of line shift cannot be always corrected as a grating movement during an integration also makes the lines broader. If you have several sky frames it may help to use a suitable combination of the sky frames (see last example).
The values of "factor" and "shift" used are stored in the output frame as descriptors IRS_FACTOR, IRS_SHIFT. Use READ/DESC to access them.

**Examples:**
```
SKYS/IRS a0012 a0013 subtr1 0 cuts=-50,50
```
defines automatically "factor" and "shift"

```
SKYS/IRS a0018 a0019 subtr2 1,0 cuts=n
```
straight subtraction + cleaning and flat-field, works silently

```
SKYS/IRS (a0101+a0104)/2 a0103*0.6+a0104*0.4 -1,0 f=1 sky=tabsky
```
example of the use of arithmetic expressions. For more information use TUTORIAL/SKYSUB

# STANDARD/IRSPEC

## STANDARD/IRSPEC *stdred/irspec* 23-SEP-1992 EO

**Purpose:** To a create a table containing a properly sampled F(lambda) vs. lambda representation of the continuum of a standard star. The user must provide an ascii file with his/her favourite guesses of the star fluxes at a number of wavelengths, e.g. the values of F(lambda) at the effective wavelengths of photometric filters in case of photometric standard stars. The output table is necessary to create the response frames (see RESPONSE/IRSPEC). You are also kindly invited to read the notes below.

**Syntax:** `STANDARD/IRSPEC in_ascii_file out_table interp_method`

`[degree=degree] [step=wavelength_step]`

`[limits=wl1,wl2] [units=wavelength_units]`

`[plot=plot_option]`

`in_ascii_file` Input, ascii (i.e. "normal") file created by the user and containing his/her best guesses of the star fluxes at a number of wavelengths (see notes below). The file must contain, in each lines, "wavelength" "flux" "any comment you like" and must not contain empty/comment rows; the format and wavelength order is free.
Example:
1.25 1.04e3 J=6.12, flux in 1e-11 erg cm-2 s-1 um-1
2.20 196 K=5.73, same units
1.65 4.34e2 H=5.99, same units

`out_table` Output table containing wavelengths (column :wl) and fluxes (column :flux) interpolated over the input values according to the method defined by the next parameter. The wavelength step and limits of the table are controlled by the "limits=.." and "step=..." options (see below).

`interp_method` Method for interpolation/fitting of input data. - b black-body fitting - p polynomial fitting, in the log(F) vs. log(wl) plane - s spline interpolation, in the log-log plane The spline method should be used only when you have many points in your input file, for few data it should give the same result as polynomial interpolation.

Additional parameters and options

`degree` degree of the polynomial/spline, default=2 Not used for black body fitting

`step` In the output table the star flux is sampled at wavelength intervals equal to "step". Default value is 0.01, be careful to properly set this parameter if you are working in Angstroms!

`limits` The output table will cover the full wavelength range specified by the input ascii files. Use this parameter if you need to restrict the size of the table (e.g. limits=1.1,1.3 or l=1.9,2.4). This does not affect the fitting procedure which always considers all the input data.

`units` Wavelengths are usually in microns. You can use units=a (or just u=a) if you prefer to work in Angstroms; your ascii file must also contain wavelengths in Angstroms and ... be careful with the step parameter (see above)

`plot` Plot option, if equal to 1 (the default) the command will plot your input points and the fitted function, Use plot=0 (or p=0) for silent work.

# Appendix K

# Standard Reduction - long

## TUTORIAL/CALIBR

*stdred/irspec* 30-NOV-93 PB

**Purpose:** Loads demonstration files and execute an interactive procedure

**Syntax:** `TUTORIAL/CALIBR`

**See also:** TUTORIAL/IRSPEC, TUTORIAL/SKYSUB

**Note:** None

**Examples:** `TUTORIAL/CALIBR`

## TUTORIAL/IRSPEC

*stdred/irspec* 30-NOV-93 PB

**Purpose:** Load demonstration files from MID_TEST directory and execute a non interactive procedure.

**Syntax:** `TUTORIAL/IRSPEC`

**See also:** TUTORIAL/CALIB, TUTORIAL/SKYSUB

**Note:** None

**Examples:** `TUTORIAL/IRSPEC`

## TUTORIAL/SKYSUB

*stdred/irspec* 30-NOV-93 PB

**Purpose:** Load demonstration files and execute an interactive procedure of sky subtraction

**Syntax:** `TUTORIAL/SKYSUB`

**See also:** TUTORIAL/IRSPEC, TUTORIAL/CALIB

**Note:** None

**Examples:** `TUTORIAL/SKYSUB`

# SUBTRACT/IRSPEC

## SUBTRACT/IRSPEC *stdred/irspec* 22-SEP-1992 EO

**Purpose:** To subtract, row by row, a polynomial fit of the continuum from an image. The user can exclude one or more areas from the fit. An image containing the fitted continua can be also produced, if requested.

**Syntax:** `SUBTRACT/IRSPEC in_ima out_ima degree [exclude=area_to_exclude]`

`[cont=continuum_image] [load=load_option]`

`in_ima` Name of the input image. Can be either 2D or 1D, untouched on exit.

`out_ima` Name of the output, continuum subtracted image. It will have the same size and limits as in_ima.

`degree` Degree of the interpolating polynomials.

Additional parameters and options.

`area_to_exclude`
Defines the region(s) to exclude from the fit (e.g. regions containing strong lines). You have three possibilities:
- CURSOR (the default) will ask you to define with the cursor the area(s) to exclude;
- [xs,ys:xe,ye] (or [xs:xe] for 1D images) defines a given area of the image (see HELP EXTR/IMA if you are not familiar with this syntax);
- table_name,tbl (e.g. mytable,tbl) in case the regions to exclude are stored in a table; you may want to create it with the cursor using GET/CURSOR table_name ? ? 999,2
for 2D images and
GET/GCURS table_name
for 1D images.

`continuum_image`
Optional name of the image containing the fitted continua, defaulted to - none - (no image created).

`load_option` 0,1 default=1. If =0 the command works silently. if =1 (default) it will display the image before and after the continuum subtraction.

**Note:** Working with 2D images be careful not to exclude too many pixels, you may get funny results. The program may get crazy if you exclude one or more entire rows.

**Examples:**
```
SUBTR/IRS cc0012 cc12sub 3

SUBTR/IRS a0118 a118sub 2 e=[@12,@28:@27,@33] c=a118cont

SUBTR/IRS iras12 iras12nc 3 l=0 c=iras12c e=line2,tab
```

**Note:** The ideal standard star should be featurless - to allow a proper cancellation of the telluric, absorption features (remember that you should not smooth the star spectrum in the IR) - and with a precisely known flux distribution. In practice, no such a star exists not only because all stars - even the hottest O types - have some absorption features, but mainly because it is hard to find accurate measurements (at fairly high resolving powers) of the absolute flux distribution of any star. At present (end 1992), there is nothing like a list of spectrophotometric standard stars as you find in the optical. Hence you must choose your reference stars according to your preferences - I personally like early O types though they are rare, little studied and often variable! - and also use your favourite guesses for their absolute fluxes at a number of wavelengths. In practice, you will probably end up using photometric standard stars in which case you can enter, in your ascii file, the values of F(lambda) at the effective wavelengths of the filters; and for the transformation from magnitudes to fluxes you may use e.g. Wilson et al. (1972, ApJ 177, 533) and Bessel (1979, P.A.S.P. 91, 589) or any other reference you like.

This command does not extrapolate beyond the wavelength limits given in the input, ascii file. If you have spectra at wavelength shorter than 1.25 micron (the effective wavelength of the J filter) you may find it difficult to estimate the star flux at wavelengths around 1 micron, and you will probably be forced to use the R - or even the V - photometric point. Be aware that interpolating over such a broad range may introduce not negligible errors!

You are recommended to use F(lambda) units rather than F(nu). However, you can also use the latter but be aware that the black-body interpolation will not work.

**Examples:**

```
STAND/IRSP bs1933.flux bs1933 bb

STAND/IRSP hd33312.flux hd33312k poly d=3 p=0 s=.001 l=1.95,2.50
```

# Standard Reduction Commands

# Context: long

# APPLY/DISPERSION

## APPLY/DISPERSION *stdred/long* PB 23-MAR-93

**Purpose:** The dispersion relation is taken from the coefficients table coerbr.tbl and applied to a 1D spectrum or a given row of a long-slit spectrum. The output spectrum is presented as a table in order to avoid interpolations.

**Syntax:** `APPLY/DISPERSION in out [y] [coef]`

| | |
|---|---|
| `in` | Name of the input image |
| `out` | Name of the output table |
| `y` | Spectrum position. The position can be provided in pixels (if preceded by an @ sign) or in world coordinates. If the input image is a 1D spectrum, the dispersion relation will be read at position y in the coefficients table. Default : @YSTART |
| `coef` | Name of the coefficients table.<br>Session keyword : COERBR<br>Default : coerbr.tbl |

**See also:** CALIBRATE/LONG, REBIN/LONG, RECTIFY/LONG, PLOT/SPECTRUM

**Note:** 1) Command CALIBRATE/LONG must have been executed previously to generate the coefficients table.
2) Command PLOT/SPECTRUM can be used to visualize the output table

**Examples:**
```
APPLY/DISPERSION ccd0001 sptab @300

PLOT/SPECTRUM sptab
```

## BATCH/LONG *stdred/long* PB 24-SEP-93

**Purpose:** Starts the batch reduction graphical user interface.

**Syntax:** `BATCH/LONG`

**See also:** PREPARE/LONG, REDUCE/INIT, REDUCE/SAVE, REDUCE/LONG

**Note:** None

**Examples:**
```
BATCH/LONG
```

## CALIBRA/FLUX *stdred/long* PB 18-NOV-94

**Purpose:** An extracted, rebinned, extinction corrected spectrum is flux calibrated by multiplication by the instrumental response function.

**Syntax:** `CALIBRA/FLUX in out [resp]`

| | |
|---|---|
| `in` | Input spectrum. It must be a 1D image in wavelength space. |
| `out` | Output flux calibrated spectrum. |
| `response` | Response image, as supplied by RESPONSE/LONG.<br>Session keyword : RESPONSE Default: response.bdf |

**See also:** EXTINCTION/LONG, INTEGRATE/LONG, RESPONSE/LONG, RESPONSE/FILTER, PLOT/FLUX, PLOT/RESPONSE

**Note:** 1) The spectrum must be corrected for the atmospheric extinction (command EXTINCTION/LONG) before flux calibration.
2) Before applying the response function, the spectra are normalised to an exposure time of 1 second. The exposure time is taken from the descriptor O_TIME(7) (in seconds).

**Examples:** `CALIBRA/FLUX spec1 flux1`

# CALIBRATE/LONG

## CALIBRATE/LONG *stdred/long* PB 20-APR-94

**Purpose:** Precise purpose of the command

**Syntax:** `CALIBRATE/LONG [tol] [deg] [mtd] [guess]`

| | |
|---|---|
| `tol` | tolerance for rejection of outliers. If tol > 0, value is assumed to be in pixel units If tol < 0, value is assumed to be in wavelength units. Session keyword: TOL; default: 0.2 |
| `deg` | Degree of the bivariate polynomial fitted to the dispersion relation. This relation is computed only if the session keyword TWODOPT is set to YES. Session keyword: DCX; default: 3,1 |
| `mtd` | mode of operation as IDENT (default), GUESS or LINEAR. In mode IDENT, the command IDENT/LONG must have been used previously. In mode GUESS, a calibrated session must have been saved by command SAVE/LONG. The mode LINEAR must be used after the command ESTIMATE/DISPERSION. This command avoids the interactive identification of lines. Session keyword: WLCMTD; default: IDENT |
| `guess` | name of reference session, as used in command SAVE/LONG. This parameter is required if mtd=GUESS. Default is the current line table defined by the keyword LINTAB. Session keyword: GUESS; no default value. |

**See also:** SEARCH/LONG, IDENT/LONG, ESTIMATE/DISPERSION, PLOT/RESIDUAL, PLOT/IDENT, PLOT/DISTORTION

**Note:** 1) The spectrum must be such that the dispersion dispersion is parallel to the rows of the image and the wavelength must be increasing from the left to the right. Start and step descriptors of the spectrum should preferably be set to 1,1.

2) Parameters relative to this command are displayed by SHO/LONG W

3) The following session keywords are involved in the wavelength calibration process: TOL, DCX, LINTAB, LINCAT, IMIN, WLCMTD, ALPHA, WLCNITER, COERBR, YSTART, GUESS, MAXDEV, SHIFT, COROPT, CORVISU, TWODOPT, START, STEP, WSTART, WEND, WSTEP.

4) Only the keywords TOL, DCX, WLCMTD, GUESS can be provided on the command line. All other keywords must be set with the command SET/LONG.

5) Session keywords WSTART, WEND, WSTEP are initialised at the end of the calibration process. These values are displayed by SHO/LONG R.

The dispersion relation is modelled independently for each row by a 1D polynomial. Interactive line identification has to be previously performed by command IDENT/LONG, on the row YSTART of the calibration image, which stores the identifications in table LINE generated by SEARCH/LONG.

Line matching in a line catalogue is then enabled in order to identify as many lines as possible and the dispersion coefficients are calculated for the row YSTART. The process goes then to the bottom and the top of the frame and uses the dispersion coefficients of each row to compute a guess for the line identifications of the following one.

The following session parameters are used:
YSTART: row on which the interactive line identification has been done
DCX(1): degree of the 1D polynomial
LINCAT: reference catalogue with comparison lines
TOL : tolerance window to assign the wavelength identification to a given line position.

The resulting dispersion coefficients are stored in the table coerbr.tbl

**Examples:** `CALIBRATE/LONG`

```
CALIBRATE/LONG deg=2,2 tol=-0.1
```

---

## CALIBRATE/TWICE *stdred/long* PB 11-JUN-93

**Purpose:** Applies twice the wavelength calibration with CALIBRATE/LONG. A selection is performed in between, which retains for the second run only those lines which were identified in all rows.

**Syntax:** `CALIBRATE/TWICE`

**See also:** SELECT/LINE, CALIBRATE/LONG

**Note:** The second calibration is performed in mode GUESS. The first one is saved in a session tmp.

**Examples:** `CALIBRATE/TWICE`

# CLEAN/LONG

## CLEAN/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Clear context Long and removes temporary tables

**Syntax:** `CLEAN/LONG`

**See also:** SET/CONTEXT, CLEAR/CONTEXT

**Examples:** `CLEAN/LONG`

## COMBINE/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Apply the command AVERAGE/IMAGE in mode AVERAGE or MEDIAN to generate a combined output image.

**Syntax:** `COMBINE/LONG catalog output [mtd]`

| | |
|---|---|
| `catalog` | Name of input catalog |
| `output` | Name of output frame |
| `method` | Combination method (AVERAGE or MEDIAN). Session keyword : COMET; default : MEDIAN |

**See also:** AVERAGE/IMAGE

**Examples:** `COMBINE/LONG mycat avflat`

## EDIT/FLUX *stdred/long* PB 02-AUG-93

**Purpose:** The instrumental response table results from INTEGRATE/LONG. Before fitting (command RESPONSE/LONG), the instrumental response table can be plotted in a graphic window and interactively modified.

**Syntax:** `EDIT/FLUX [resp]`

| | |
|---|---|
| `resp` | Name of the response table.<br>Session keyword: RESPTAB; default: resp.tbl |

**See also:** EXTINCTION/LONG, INTEGRATE/LONG, RESPONSE/LONG, RESPONSE/FILTER, CALIBRATE/FLUX, PLOT/RESPONSE, PLOT/FLUX.

**Note:** None.

**Examples:** `EDIT/FLUX`

## ERASE/LONG *stdred/long* PB 02-AUG-93

**Purpose:** Calibrated line positions (resulting from CALIBRATE/LONG) are plotted in a pixel-wavelength space. By mean of the graphic cursor, positions can be interactively rejected. A new calibration can then be performed.

**Syntax:** `ERASE/LONG`

**See also:** CALIBRATE/LONG, PLOT/RESIDUAL, PLOT/CALIBRATE, PLOT/DELTA, CALIBRATE/TWICE, PLOT/DISTORTION

**Note:** None

**Examples:** `ERASE/LONG`

## ESTIMATE/DISPERSION *stdred/long* PB 11-APR-93

**Purpose:** Using an approximate definition of the central wavelength and average dispersion, the command provides a better estimate of the linear model, allowing to start the command CALIBRATE/LONG in mode WLCMTD=LINEAR. This new command is provided on an experimental basis. Any feedback is appreciated.

**Syntax:** `ESTIMATE/DISPERSION wdisp wcent [ystart] [line] [cat]`

| | |
|---|---|
| `wdisp` | dispersion,tol,acc. The dispersion is provided in wavelength units per pixel. The tolerance and the accuracy are in percent (Defaults: 30%, 1%). |
| `wcent` | central wavelength,tol,acc. The central wavelength is in wavelength units (same units as in the line catalog LINCAT). Tolerance and accuracy are in pixels. (Defaults: 100, 1). |
| `ystart` | Starting row, in pixels. The value is automatically set to the central row number if YSTART = 0<br>Session keyword: YSTART; default: 0 |
| `line` | Name of the line table.<br>Session keyword: LINTAB; default: line.tbl |
| `line` | Name of the line catalog.<br>Session keyword: LINCAT; no default value. |

**See also:** SEARCH/LONG, CALIBRATE/LONG

**Note:** 1) Since the method is based on a linear model of the dispersion relation, the results can be inaccurate if the dispersion relation presents an important error of linearity. This is normally the case with the Boller and Chivens spectrograph.

**Examples:** `ESTIMATE/DISPERSION wcent=5200 wdisp=2.1`

# EXTINCTION/LONG

## EXTINCTION/LONG *stdred/long* 29-MAR-1993 PB

**Purpose:** Correct spectra for interstellar or atmopsheric extinction.

**Subject:** Spectroscopy, Calibration, Extinction.

**Syntax:** `EXTINCTION/LONG in out [scale] [table] [col]`

| | |
|---|---|
| `in` | the frame to be corrected for extinction. It must be a 1D rebinned spectrum. |
| `out` | the frame to hold the corrected data |
| `scale` | the scaling factor:<br>a) E(B-V) if interstellar reddening<br>b) AIRMASS if atmospheric extinction.<br>If scale=0, the scale will be defaulted to the AIRMASS descriptors from the frame.<br>Otherwise, the indicated value will be taken as airmasss.<br>Default: 0 |
| `table` | the table with the wavelength dependence of the reddening or the atmospheric extinction.<br>Session keyword: EXTAB; default: atmoexan.tbl |
| `col` | Number of the transmission column in the atmospheric extinction table; default: 2 |

**Note:** The table supported by MIDAS to correct for interstellar extinction is:

1a) INSTEXAN contains in column #1 the wavelength in Angstroem units and in columns 2-8 seven different extinction laws the references for which are as follows:
col. 2: Savage, B.D., Mathis, J.S., Ann. Rev. Astron. Astrophys. No. 17, 73 (1979): Galaxy
col. 3: Nandy et al., MNRAS Vol. 196, 955 (1981): LMC
col. 4: Prevot et al., Astron. Astrophys. Vol. 132, 389 (1984): SMC
col. 5: Koornneef, J., Code, A.D., ApJ Vol. 247, 860 (1981): LMC
col. 6: Seaton, M.J., MNRAS Vol. 187, 73P (1979): Galaxy fit
col. 7: Howarth, I.D., MNRAS Vol. 203, 301 (1983): Galaxy fit
col. 8: Howarth, I.D., MNRAS Vol. 203, 301 (1983): LMC fit

All laws are normalized to A/E(B-V) = 3.1 at 5550 A and interpolated and rebinned to constant step in wavelength (10 A). Note that the wavelength coverage is very different for the various data sets.

1b) The contents of INSTEXNM is the same except for use with spectra calibrated in nanometers

Correction for atmospheric extinction is possible with:

2a) atmoexan.tbl containing the wavelength in Angstroem units in column #1 and the extinction law for La Silla as described by H. Tug in the ESO Messenger No.11, December 1977, and listed in the ESO User's Manual.
2b) ATMOSNM which is identical to atmoexan.tbl except for wavelengths in nm

If "scale" is defaulted, descriptor "AIRMASS" is read from frame "inframe".

These tables are stored in MID_EXTINCTION:

**Examples:** `EXTINCTION/LONG observed corrected`

## EXTRACT/AVERAGE *stdred/long* PB 23-MAR-93

**Purpose:** Invokes command AVERAGE/ROW to estimate the sky and the object spectrum within defined limits and subtracts the sky from the object.

**Syntax:** `EXTRACT/AVERAGE in out [obj] [sky] [mtd]`

| | |
|---|---|
| `in` | Input long-slit spectrum |
| `out` | Output one-dimensional spectrum |
| `obj` | low,upp. Lower and upper limits of the object spectrum, in pixels. Session keyword: OBJECT; default: 0,0 |
| `sky` | low1,upp1,low2,upp2. Lower and upper limits of the sky. Session keywords: LOWSKY, UPPSKY; default: 0,0,0,0 |
| `mtd` | Extraction method (LINEAR or AVERAGE). Session keyword: EXTMTD; default: LINEAR |

**See also:** AVERAGE/ROW, EXTRACT/LONG, GET/CURSOR

**Note:** It is assumed that :
LOWSKY(1)<=LOWSKY(2) < OBJECT(1)<=OBJECT(2) < UPPSKY(1)<=UPPSKY(2)

If method LINEAR is used, the averaged flux of the sky subtracted object is multiplied by the width of the object.

Parameters relative to this command are displayed by
SHO/LONG E

The command creates temporary images in the following files:
Lower sky : middumma.bdf
Upper sky : middummb.bdf
Average sky : middumms.bdf
Average object : middummo.bdf

**Examples:**
```
EXTRACT/AVERAGE ccd0001 spec1 190,192 180,188,194,202
```

# EXTRACT/LONG

## EXTRACT/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Extract spectrum from CCD frame. The rows of the spectrum are added with weights which are chosen for optimal S/N-ratio of the resulting spectrum. Cosmic ray hits are removed by analyzing the profile perpendicular to the dispersion (assumed to be along the x-axis). The method is only suited for spectra of (spatially) unresolved sources. The method is very similar to the one described by Horne (1986, PASP 98, 609).

**Syntax:** `EXTRACT/LONG in out [sky] [obj] [order,niter] [ron,g,sigma]`

| | |
|---|---|
| `in` | Input long-slit spectrum |
| `out` | Name of the output 1D spectrum |
| `sky` | a) frame with fitted sky spectrum<br>(e.g., as supplied by command SKYFIT/LONG)<br>= b) constant (e.g. 0, if no sky shall be subtracted); Default : 0. |
| `obj` | low,upp. Boundaries (in y) of the object spectrum,<br>in pixels.<br>Session keyword : OBJECT; no default value. |
| `order,niter` | order of polynomial fitted to spatial profile,<br>no. of iterations.<br>Session keywords : ORDER, NITER; default : 3,3 |
| `ron,g,sigma` | read-out-noise (ADU), inverse gain factor (e-/ADU), threshold for the removal of cosmic ray hits (in units of the standard deviation calculated for each pixel from the number of electrons and the ron)<br>Session keywords : RON, GAIN, SIGMA<br>Defaults : 7, 2, 3 |

**See also:** SKYFIT/LONG, EXTRACT/AVERAGE, APPLY/DISPERSION

**Note:** a) In order for the weighting procedure to work properly, the ADU numbers must not differ significantly from the raw ones after bias correction. Therefore, any prior flat-fielding requires usage of a flat-field normalised to its mean flux (command NORMALIZE/FLAT).

b) Although the extraction must normally be performed after spectrum rebinning (using REBIN/LONG or RECTIFY/LONG), it can also be performed on non-rebinned spectra.

**Examples:** `EXTRACT/LONG inframe result skyframe 251,263 3,3 36,6,4`

## GCOORD/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Invokes command GET/CURSOR to get coordinates from the display window.

**Syntax:** `GCOORD/LONG [number] [outtab]`

| | |
|---|---|
| `number` | Number of coordinates; default : 1 |
| `outtab` | Name of the output table.<br>Session keyword : COORFIL; default: middummgc.tbl |

**See also:** MAKE/DISPLAY, GET/CURSOR, GRAPH/LONG

**Note:** None

**Examples:** `GCOORD/LONG`

## GRAPH/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Invokes CREATE/GRAPH and SET/GCURSOR to generates a large size graphic window.

**Syntax:** `GRAPH/LONG [size] [position] [id]`

| | |
|---|---|
| `size` | Window size. Default: 1150,350 |
| `position` | Window position. Default: 1,1 |
| `id` | Window id. Default : 0 |

**See also:** CREATE/GRAPH, SET/GCURSOR, MAKE/DISPLAY

**Note:** None

**Examples:** `GRAPH/LONG`

## HELP/LONG *stdred/long* PB 23-MAR-93

**Purpose:** provides information about session keywords.

**Syntax:** `HELP/LONG [keyword]`

| | |
|---|---|
| `keyword` | session keyword name. The full list of keywords is displayed with SHOW/LONG |

**See also:** SHOW/LONG, INIT/LONG, SAVE/LONG, SET/LONG

**Note:** None

**Examples:** `HELP/LONG wlc`

# IDENTIFY/LONG

## IDENTIFY/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Plots a row of the arc spectrum and invokes command GET/GCURSOR for intercative identification of lines.

**Syntax:** `IDENT/LONG [wlc] [ystart] [lintab] [tol]`

| | |
|---|---|
| `wlc` | Wavelength calibration frame Session keyword : WLC; no default value |
| `ystart` | Starting row number (pixels) for identification and calibration. Session keyword : YSTART; default : 0 |
| `lintab` | Line table, as generated by command SEARCH/LONG Session keyword : LINTAB; default : line.tbl |
| `tol` | Tolerance for the identification of lines, in world coordinates. This parameter must be set if the interactive pointing generates too often the error message: "Feature not found, try again..." (No session keywords controls this parameter) Default: 4. |

**See also:** SEARCH/LONG, GRAPH/LONG, CALIBRATE/LONG, CUTS, SET/GRAPH, ESTIMATE/DISPER

**Note:** 1) This command requires two preliminary actions:
- Creation of a line table with SEARCH/LONG
- Creation of a graphic window with GRAPH/LONG

2) If YSTART=0, its value is automatically set to the central row number.

3) The arc spectrum is plotted with respect to the current set-up of plotting parameters. To adjust the plotting, use commands CUTS or SET/GRAPH.

4) Interactive identification of lines can be avoided by the command ESTIMATE/DISPERSION. This command is however provided on an experimental basis.

**Examples:** `IDENT/LONG ystart=200`

## INITIALIZE/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Creation and initialisation of session keywords to default values or to previously saved values (see command SAVE/LONG).

**Syntax:** `INITIALIZE/LONG [session]`

| | |
|---|---|
| `session` | Name of a session previously saved with command SAVE/LONG |

**See also:** SAVE/LONG, SHOW/LONG, HELP/LONG

**Note:** None

**Examples:** `INITIALIZE/LONG`

## INTEGRATE/LONG *stdred/long* PB 01-APR-93

**Purpose:** The reduced spectrum of a standard star (after extraction and rebinning) is compared to a flux table to generate the response table. This table must be processed further with RESPONSE/LONG to generate the final response image.

**Syntax:** `INTEGRATE/LONG std [flux] [resp]`

| | |
|---|---|
| `std` | one-dimensional reduced spectrum of a standard star. |
| `flux` | Flux table of the standard star. A collection of flux tables is available in MID_FLUX.<br>Session keyword : FLUXTAB; no default value |
| `resp` | Output response table. Session keyword : RESPTAB; default: resp.tbl |

**See also:** EXTINCTION/LONG, RESPONSE/LONG, RESPONSE/FILTER, PLOT/FLUX, PLOT/RESPONSE, CALIBRATE/FLUX

**Note:** None

**Examples:** `INTEGRATE/LONG sp23 1745x4`

## LINADD/LONG *stdred/long* PB 01-APR-93

**Purpose:** Adds entries to the table line.tbl. The command is similar to SEARCH/LONG.

**Syntax:** `LINADD/LONG in w,bin [y] [mtd] [line] [out]`

| | |
|---|---|
| `in` | Input image |
| `w,bin` | window, bin size |
| `y` | Starting line, in pixels.<br>Session keyword : YSTART; initialised by SEARCH/lONG |
| `seamtd` | Search method.<br>Session keyword : SEAMTD; default: GAUSSIAN |
| `line` | line table.<br>Session keyword : LINTAB; default: line.tbl |
| `out` | output modified line table; default: lineadd.tbl |

**See also:** SEARCH/LONG

**Note:** None

**Examples:** `LINADD/LONG ccd0023 5,5 out=line2.tbl`

# LOAD/LONG

## LOAD/LONG *stdred/long* PB 02-AUG-93

**Purpose:** Compute CUTS values and load an image in the display window.

**Syntax:** `LOAD/LONG image [scale_x,[scale_y]]`

`image` Image name.

`scale_x,scale_y` Scaling factor on x- and y-axis.

**See also:** CUTS, LOAD/IMAGE

**Note:** This command is based on the IHAP algorithm of cuts computation.

**Examples:** `LOAD/LONG ccd0001 2,2`

## MAKE/DISPLAY *stdred/long* PB 23-MAR-93

**Purpose:** If there is no existing display window ,the command creates one with default parameters and loads the heat look-up table.

**Syntax:** `MAKE/DISPLAY`

**Examples:** `MAKE/DISPLAY`

## NORMALIZE/FLAT *stdred/long* PB 1-AUG-95

**Purpose:** The flat-field is divided by its average shape, obtained by fitting a polynomial along the dispersion direction of an averaged flat-field.

**Syntax:** `NORMALIZE/FLAT in out [bias] [deg] [fit] [visu]`

| | |
|---|---|
| `in` | input image, consisting of a flat-field or a combination of flat-fields (resulting from COMBINE/LONG). |
| `out` | Name of the output normalised flat-field. |
| `bias` | bias image or constant to be subtracted prior to normalisation.<br>Session keyword: BIAS; default: 0 |
| `deg` | Order of the polynomial.<br>Session keyword: FDEG; default: 2 |
| `fit` | fitted 1D spectrum used for normalisation.<br>Session keyword: FFIT; default: middummf.bdf |
| `visu` | Visualisation flag (YES/NO). If visu=YES, the central row of the flat field is plotted as well as the fitted function.<br>Session keyword: FVISU; default: YES |

**See also:** COMBINE/LONG, CONTINUUM/SPEC

**Note:** 1) Session keywords relative to this command are displayed by SHOW/LONG D

2) Smoothing spline fitting can be performed with the command CONTINUUM/SPEC.

**Examples:** `NORMALIZE/FLAT flat ffnorm deg=3 visu=no`

## PLOT/CALIBRATE *stdred/long* PB 23-MAR-93

**Purpose:** This command allows to plot the arc lines identified by CALIBRATE/LONG. The lines can also interactively be rejected from calibration or their position read with the graphic cursor.

**Syntax:** `PLOT/CALIBRATE [mode]`

| | |
|---|---|
| `mode` | interaction mode. Possible values: NONE, EDIT, CURSOR. Default : NONE |

**See also:** PLOT/IDENT, PLOT/RESIDUAL, PLOT/DISTORTION, PLOT/DELTA

**Note:** The arc spectrum is plotted in the graphic window. The plotted row is controlled by the session keyword YSTART.

**Examples:** `PLOT/CALIBRATE`

# PLOT/DELTA

## PLOT/DELTA *stdred/long* PB 11-JUN-93

**Purpose:** This command plots the dispersion relation resulting from CALIBRATE/LONG in a pixel-wavelength space. The arc lines used for the fit are also plotted and can be rejected by mean of the graphic cursor in mode EDIT.

**Syntax:** `PLOT/DELTA [mode]`

| | |
|---|---|
| `mode` | Interaction mode. Possible values: NONE, EDIT. Default: NONE. |

**See also:** PLOT/CALIB, CALIBRATE/LONG, PLOT/RESIDUAL

**Note:** None

**Examples:**
```
PLOT/DELTA edit
```

## PLOT/DISTORTION *stdred/long* PB 11-JUN-93

**Purpose:** The fitted positions of the line catalog are plotted in wavelength/y-coordinate space. The commands allows to check the regularity of the fitted dispersion relation along the y-axis. Both row-by-row and bivariate solutions can be verified depending on the mode.

**Syntax:** `PLOT/DISTORTION wave [delta] [mode]`

| | |
|---|---|
| `wave` | Arc line wavelength. The wavelength must be chosen from the line catalog (session keyword LINCAT). |
| `delta` | Half-range in pixels of the wavelength space. Default value is 0.5, which means that the full graphic window represents one pixel on the spectrum. |
| `mode` | Dispersion relation mode. Possible values: RBR, 2D Allows to plot the row-by-row (RBR) or the bivariate (2D) solution. Default is RBR. |

**See also:** CALIBRATE/LONG, PLOT/DELTA, PLOT/RESIDUAL, PLOT/CALIBRATE

**Note:** mode = 2D can only be used if the session keyword TWODOPT is set to YES (i.e. if a bivariate dispersion solution has been estimated).

**Examples:**
```
PLOT/DISTORTION 5008.645 0.1 2D
```

## PLOT/FLUX *stdred/long* PB 23-MAR-93

**Purpose:** Plots the flux table

**Syntax:** `PLOT/FLUX [fluxtab]`

| | |
|---|---|
| `fluxtab` | Standard star flux table<br>Session keyword : FLUXTAB; no default value. |

**See also:** INTEGRATE/LONG, RESPONSE/LONG, PLOT/RESPONSE, CALIBRATE/FLUX, RESPONSE/FILTER

**Note:** None

**Examples:** `PLOT/FLUX`

## PLOT/IDENT *stdred/long* PB 23-MAR-93

**Purpose:** Plot interactive identifications

**Syntax:** `PLOT/IDENT [wlc] [line] [x] [id] [wave]`

| | |
|---|---|
| `wlc` | Wavelength calibration frame.<br>Session keyword: WLC; no default value |
| `line` | Line table.<br>Session keyword: LINTAB; default : line.tbl |
| `x` | Column X; default: :X. |
| `id` | Column IDENT; default: :IDENT. |
| `wave` | Column WAVE; default: :WAVE. |

**See also:** PLOT/CALIBRATE, PLOT/RESIDUAL, PLOT/DISTORTION

**Note:** The arc spectrum is plotted in the graphic window. The plotted row is controlled by the session keyword YSTART.

**Examples:** `PLOT/IDENT`

# PLOT/RESIDUAL

## PLOT/RESIDUAL *stdred/long* PB 23-MAR-93

**Purpose:** Plots residuals after wavelength calibration.

**Syntax:** `PLOT/RESIDUAL [y] [table]`

| | |
|---|---|
| `y` | plotted row number (pixel value).<br>Session keyword : YSTART; default: central row. |
| `table` | Line table.<br>Session keyword : LINTAB; default : line.tbl |

**See also:** CALIBRATE/LONG

**Note:** None

**Examples:** `PLOT/RESIDUAL 120`

## PLOT/RESPONSE *stdred/long* PB 23-MAR-93

**Purpose:** Plots the response correction function

**Syntax:** `PLOT/RESPONSE [resp]`

| | |
|---|---|
| `resp` | Response image, as supplied by RESPONSE/LONG.<br>Session keyword: RESPONSE; default: response.tbl |

**See also:** PLOT/FLUX, RESPONSE/LONG, RESPONSE/FILTER, INTEGRATE/LONG

**Note:** None

**Examples:** `PLOT/RESPONSE`

## PLOT/SEARCH *stdred/long* PB 11-JUN-93

**Purpose:** Plot the position of the searched lines resulting from SEARCH/LONG in a pixel-pixel space

**Syntax:** `PLOT/SEARCH [mode] [table]`

| | |
|---|---|
| `mode` | Visualization mode (1D, 2D). In mode 1D, the procedure plots the first row of the arc spectrum (session keyword WLC) and overplots the positions of the searched lines. This mode must be used for 1D spectra.<br>In mode 2D, the positions are plotted in an x-y pixel space. |
| `table` | Input table.<br>Session keyword : LINTAB; default : line.tbl |

**See also:** SEARCH/LONG

**Note:** None

**Examples:** `PLOT/SEARCH 1D`

## PLOT/SPECTRUM *stdred/long* PB 23-MAR-93

**Purpose:** Plots a 1D spectrum in table format, as supplied by APPLY/DISPERSION

**Syntax:** `PLOT/SPECTRUM table`

| | |
|---|---|
| `table` | Name of the table resulting form APPLY/DISPERSION |

**See also:** APPLY/DISPERSION

**Note:** None

**Examples:** `PLOT/SPECTRUM`

## PREPARE/LONG *stdred/long* PB 01-APR-93

**Purpose:** Applies command EXTRACT/IMAGE to an image (parameter in is given with an extension .bdf) or to a catalog (default) and generate output images which root name is specified by out.

**Syntax:** `PREPARE/LONG in [out] [limits]`

| | |
|---|---|
| `in` | Input frame or catalog. An input frame must be identified by the extension .bdf. An input catalog can be specified with or without the extension .cat. |
| `out` | Root name for output files. Default: ext |
| `limits` | x1,y1,x2,y2. Limits of the extraction window, in pixels. x1,y1 is the position of the lower-left corner, and x2,y2 the position of the upper-right corner. Null values can be used to represent the limit of the frame.<br>Session keyword: TRIM; default: 0,0,0,0 |

**See also:** EXTRACT/IMAGE

**Note:** None

**Examples:**

```
CREATE/ICAT mycat *.bdf
PREPARE/LONG mycat trim=0,60,0,450
```

Extracts all rows from y=60 to y=450

# REBIN/LONG

## REBIN/LONG *stdred/long* PB 01-APR-93

**Purpose:** Non linear rebinning of 2D long-slit spectra using the row-by-row method.

**Syntax:** `REBIN/LONG in out [start,end,step] [mtd] [table]`

| | |
|---|---|
| `in` | Input image. |
| `out` | Output spectrum. |
| `start,end,step` | Starting, final and bin wavelength, in wavelength units.<br>Session keywords: REBSTRT, REBEND, REBSTP<br>Values initialised by CALIBRATE/LONG. |
| `mtd` | Rebinning method. (LINEAR, QUADRATIC, SPLINE).<br>Session keyword: REBMTD; default: LINEAR |
| `table` | coefficients table.<br>Session keyword: COERBR; default: coerbr.tbl |

**See also:** CALIBRATE/LONG, RECTIFY/LONG

**Note:** Very strong variations of flux, like in bright emission lines or cosmics can yield to negative interpolated values with methods QUADRATIC or SPLINE.

**Examples:** `REBIN/LONG ccd0023 reb23`

## RECTIFY/LONG *stdred/long* 20-AUG-1985 DB/KB

**Purpose:** Geometrically rectify a distorted 2-D spectrum (e.g. obtained with an image tube) and rebin it to constant step in wavelength

**Syntax:** `RECTIFY/LONG in out [reference] [nrep] [deconvol_flag]`

| | |
|---|---|
| `in` | input frame |
| `out` | output frame |
| `reference` | a) image to define the output sampling domain. b) start,step,npix to define along X axis the start and step in wavelength and the number of pixels. c) if no value is provided, start,step and npix are automatically computed. |
| `nrep` | additional sub-stepping factor (cf. "note" below) default: 1, maximum: 5 |
| `deconvol_flag` | deconvolution flag. Y=YES, N=NO. Default: N. |

**See also:** SEARCH/LONG, IDENTIFY/LONG, CALIBRATE/LONG, REBIN/LONG, APPLY/DISPERSION

**Note:** The photometric accuracy per pixel is thought to be better than about 1The flux conservation is perfect if also nearest neighbours are considered. At the expense of further CPU time consumption, the said 1This seems justified only for observations of the highest S/N. The programme routinely performs a simple 'deconvolution' assuming a fixed point spread function (PSF) of about one pixel which appears appropriate for a wide range of applications. This can be suppressed (deconvol_flag = N) for test purposes or low-resolution data, i.e. data with a FWHM of the PSF of a few pixels, but even for high-resolution data the results are not bad. With deconvol_flag = N (and nrep = 1), the execution time is reduced by a factor of 5 !
The DIFFERENCE to RECTIFY/IMAGE is that in this version the number of pixels along the "x" (wavelength) axis in output is forced to be the same as in input thus roughly preserving the identity of the resolution elements. The geometric rectification and the rebinning to constant step in wavelength are done simultaneously which avoids the degradation of the data due to the twofold rebinning that otherwise would be necessary. Tests show that with a set of carefully selected comparison lines an accuracy of the wavelength calibration of better than 0.2 pixels (RMS error) can easily be achieved with nrep = 1 and deconvol_flag='Y'.
The dispersion coefficients defining the geometric transformation are stored in the keywords KEYLONGC (character), KEYLONGI (integer) and KEYLONGD (double precision) to model the distortion in the wavelength axis, and in the keywords COEFYC (character), COEFYI (integer) and COEFYD (double precision) to model the distortion perpendicular to the wavelengths. These coefficients are generated with the command REGRESSION/POLYNOMIAL.
The resampling step (keyword REBSTP) should preferably be close to or larger than the average dispersion per pixel, as provided by the command CALIBRATE/LONG.

**Examples:**

```
>SET/LONG WLC=wlc LINCAT=hear WIDTH=5 THRES=15.
>SEARCH/LINE
>IDENTIFY/LONG
>CALIBRATE/LONG
>RECTIFY/LONG object output !  resample the object
```

This example shows how to prepare all the required information for RECTIFY/LONG.

# REDUCE/INIT

## REDUCE/INIT *stdred/long* 24-SEP-93 PB

**Purpose:** Batch reduction parameters can be saved with REDUCE/SAVE and loaded again with REDUCE/INIT

**Syntax:** `REDUCE/INIT partab`

`partab` Name of the previously saved set of reduction parameters.

**See also:** REDUCE/SAVE, PREPARE/LONG, REDUCE/LONG, BATCH/LONG

**Note:** REDUCE/INIT reads a file which name is 'partab' followed by the extension .brf

**Examples:** `REDUCE/INIT ses1`

## REDUCE/LONG *stdred/long* PB 01-APR-93

**Purpose:** Performs the follwoing steps on a catalog of long-slit raw spectra:
- Bias correction
- Dark correction
- Flat-Field correction
- Rotation
- Extraction of a sub-image
- Rebinning
- Extraction
- Flux calibration

**Syntax:** `REDUCE/LONG input`

`input` Name of the input image catalog

**See also:** CREATE/ICAT, CREATE/GUI

**Note:** This high-level command involves a great number of parameters, set by mean of the commands SHOW/LONG B, SET/LONG or by mean the batch reduction graphical interface (command CREATE/GUI LONG).

**Examples:**

```
ttCREATE/ICAT mycat *.bdf
REDUCE/LONG mycat
```

## REDUCE/SAVE *stdred/long* 24-SEP-93 PB

**Purpose:** Batch reduction parameters can be saved with REDUCE/SAVE and loaded again with REDUCE/INIT

**Syntax:** `REDUCE/SAVE partab`

| | |
|---|---|
| `partab` | Name of the set of reduction parameters. |

**See also:** REDUCE/INIT, PREPARE/LONG, REDUCE/LONG, BATCH/LONG

**Note:** REDUCE/SAVE writes a file which name is 'partab' followed by the extension .brf

**Examples:** `REDUCE/SAVE ses1`

## RESPONSE/FILTER *stdred/long* 01-APR-93 PB

**Purpose:** Compares an observed standard star spectrum to a reference flux table and smooths the resulting response by filtering

**Syntax:** `RESPONSE/FILTER std [flux] [resp]`

| | |
|---|---|
| `std` | one-dimensional reduced spectrum of a standard star. |
| `flux` | Flux table of the standard star. A collection of flux tables is available in MID_FLUX.<br>Session keyword: FLUXTAB; no default value |
| `resp` | Output response image. Session keyword: RESPONSE; default: response.bdf |

**See also:** INTEGRATE/LONG, RESPONSE/LONG, PLOT/FLUX, PLOT/RESPONSE, CALIBRATE/FLUX, EXTINCTION/LONG

**Note:** The command applies in sequence a median and a smooth filter to the ratio image. The parameters of the filters are respectively controlled by the session keywords FILTMED and FILTSMO

**Examples:** `RESPONSE/FILTER sp23 1745x4`

# RESPONSE/LONG

## RESPONSE/LONG *stdred/long* PB 23-MAR-93

**Purpose:** The response correction table resulting from INTEGRATE/LONG (and optionally edited by EDIT/FLUX) is converted to an image in wavelength-ratio or wavelength-color space. The interpolation is performed by polynomial or smoothing splines.

**Syntax:** `RESPONSE/LONG [plot] [fit] [deg] [smo] [table] [image] [visu]`

| | |
|---|---|
| `plot` | Type of plot: RATIO or MAGNITUDE<br>Session keyword: PLOTYP; default : RATIO |
| `fit` | Type of fit (POLY or SPLINE).<br>Session keyword: FITYP; default : POLY |
| `deg` | Degree of fit.<br>Session keyword: FITD; default : 3 |
| `smooth` | Smoothing factor (used only if fit=SPLINE)<br>Session keyword : SMOOTH; default : 0. |
| `table` | Response table, as supplied by INTEGRATE/LONG<br>Session keyword: RESPTAB; default : resp.tbl |
| `image` | Output response image.<br>Session keyword: RESPONSE; default : response.bdf |
| `visu` | Visualisation option: YES or NO<br>Session keyword: RESPLOT; default : YES |

**See also:** INTEGRATE/LONG, PLOT/RESPONSE, PLOT/FLUX, EXTINCTION/LONG, EDIT/FLUX, CALIBRATE/FLUX

**Note:** None.

**Examples:** `RESPONSE/LONG`

## SAVE/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Saves session keywords as descriptors of the table line.tbl and saves additional process tables.

**Syntax:** `SAVE/LONG [session]`

| | |
|---|---|
| `session` | Session name. |

**See also:** SEARCH/LONG, INIT/lONG, HELP/LONG, SHOW/LONG

**Note:** a) The command executes the follwowing operations:
- Copy the table line.tbl
- Copy the table coerbr.tbl, if it exists
- Save all keywords values in the copy of line.tbl

b) Session keywords cannot be saved if the table line.tbl does not exist. This table is created by use of the command SEARCH/LONG

**Examples:** `SAVE/LONG night1`

SEARCH/LONG *stdred/long* PB 20-APR-94

**Purpose:** search for spectral features in a long-slit spectrum

**Syntax:** `SEARCH/LONG [in] [thres] [width] [yaver] [step] [mtd] [mode]`

| | |
|---|---|
| `in` | Input image Session keyword: WLC; no default value |
| `thres` | Threshold in ADU units. The threshold must be counted above the local background. Session keyword: THRES; default: 30. |
| `width` | Width of the local histogram window (in pixels) Session keyword: WIDTH; default: 8 |
| `yaver` | Row average parameter. The actual number of rows averaged before detection is 2*yaver+1. Session keyword: YWIDTH; default: 0 |
| `step` | Step in pixels along the y-axis. Session keyword: YSTEP; default: 1 |
| `mtd` | Centering method. Possible values are GAUSSIAN, GRAVITY, MAXIMUM. Session keyword: SEAMTD; default: GAUSSIAN |
| `mode` | Type of spectral features (EMISSION or ABSORPTION) (This parameter is not controlled by a session keyword) Default: EMISSION |

**See also:** IDENTIFY/LONG, CALIBRATE/LONG

**Note:** 1: The input image is assumed to be rotated in such a way that the dispersion direction is oriented along the rows. The wavelength must be increasing from the left to the right and the STEP descriptors of the image must be positive.

2: Session keywords START, STEP, NPIX are initialised by this command.

3: Spectral features are detected along sucessive rows separated by <step>. Before detection, 2*yaver+1 rows are averaged for noise reduction.

4: Algorithm: A line detection occurs if the number of counts in a pixel is greater than the local threshold estimated as the sum of the local background and <thres>. The local background value is estimated as the local median in a window of size <width>. Multiple detections are checked and line centers are estimated by gaussian, gravity or maximum method.

5: The output table is used to store the position and peak value of the detected lines as follow: X : center of the line in world coordinates (wavelength direction) Y : scan row number (in world coordinates) PEAK : peak value of the line (ADU)

**Examples:** `SEARCH/LONG arc1`

```
SET/LONG WLC = arc1
SEARCH/LONG thres=300
```

# SELECT/LINE

## SELECT/LINE *stdred/long* PB 11-JUN-93

**Purpose:** For each row position of the table line.tbl (session keyword LINTAB), the command CALIBRATE/LONG identifies lines and fits a dispersion relation. The command SELECT/LINE selects those lines identified for all row positions. SELECT/LINE is a low-level command invoked by CALIBRATE/TWICE.

**Syntax:** `SELECT/LINE`

**See also:** CALIBRATE/TWICE, CALIBRATE/LONG

**Note:** None

**Examples:** `SELECT/LINE`

## SET/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Assigns a value to long-slit session keywords

**Syntax:** `SET/LONG key=value [...]`

| | |
|---|---|
| `key` | keyword name |
| `value` | parameter value |

**See also:** SHOW/LONG, HELP/LONG, SAVE/LONG, INIT/LONG

**Note:** a) Space characters can be inserted on the right or left part of the sign "=". The name of the echelle keyword can be truncated to a non ambiguous root.

b) String values can be specified between double quotes. The double quote (") is a reserved character and cannot be used in a value.

c) Short description of long-slit keywords is provided by HELP/LONG command, values are displayed by SHOW/LONG.

d) It is possible to assign a value to long-slit keywords using monitor assignment. In this case, there must be a space on both parts of the sign =. The syntax depends on the definition of the keyword (provided by HELP/LONG). Examples:
FLAT = "ff456.bdf" (keyw. FLAT/C/1/60)
WIDTH = 12 (keyw. WIDTH/I/1/1)
OBJECT(2) = 252 (keyw. OBJECT/I/1/2)
To assign values to character keywords, additional spaces must be inserted to overwrite a longer previous value.

e) To assign the value of an element of a multiple element keyword (e.g. OBJECT/I/1/2) it is possible to use the direct assignment (OBJECT(2) = 252) or the SET/LONG by inserting comas in place of the values that must not be changed (SET/LONG OBJ = ,252).

f) The total length of the line is limited to 132 characters.

**Examples:** `SET/LONG FLAT =ff456.bdf WIDTH=12 OBJECT = ,252 INSTR = "EFOSC 2"`

SHOW/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Displays values of session keywords.

**Syntax:** `SHOW/LONG [section]`

`section` Section to be listed. Default is ALL.
Possible values are:
G) General values W) Wavelength calibration D) Dark, Bias and Flat-Fields E) Extraction R) Rebin F) Flux calibration B) Batch reduction A) All parameters

**See also:** SET/LONG, INIT/LONG, SAVE/LONG, HELP/LONG

**Note:** a) The information displayed depends on the chosen option so that only required parameters are displayed. For example try:
SET/LONG WLCM=IDENT
SHO/LONG W
SET/LONG WLCM=GUESS
SHO/LONG W

b) The list of parameters is by default presented by successive screens. An uninterrupted list is displayed if the keyword SESSDISP is set to NO:
SESSDISP = "NO " (Default is "YES")

**Examples:** `SHOW/LONG F`

# SKYFIT/LONG

## SKYFIT/LONG *stdred/long* 01-APR-1993 PB

**Purpose:** Fit a polynomial to the data in two windows along the y-axis. Two modes can be selected:

MODE 0: It is assumed, that the normalized spatial profile is constant with x, i.e. only one polynomial is fitted to the mean spatial profile and in all columns scaled to the actual flux. MODE 1: For every column, an independent polynomial is fitted.

In either mode, the fitted polynomial(s) is (are) evaluated for every increment in x. The output frame contains the evaluation of the polynomial(s) all along the associated column(s), not only in the windows used for the fitting. The command is designed to provide a frame that approximates the sky contamination of spectra at the y-position of the object.

**Syntax:** `SKYFIT/LONG input output [sky] [degree] [mode] [g,r,t] [radius]`

| | |
|---|---|
| `input` | input image |
| `output` | output image (sampled as the input spectrum) |
| `sky` | sky limits in pixel numbers, for ranges a1 to a2 and b1 to b2 to be entered as a1,a2,b1,b2.<br>Session keywords: LOWSKY, UPPSKY; default: 0,0 |
| `degree` | order of the polynomial fit.<br>Session keyword: SKYORD; default: 1 |
| `mode` | same spatial profile for all columns (0) or independent profile for every column (1)<br>Session keyword: SKYMOD; default: 0 |
| `g,r,t` | gain, ron, threshold for cosmic rays rejection. (used only if mode=0). gain is in e-/ADU, ron in ADU, threshold in units of the standard deviation. Session keywords: GAIN, RON, SIGMA; defaults: 2, 7, 3 |
| `radius` | Radius for cosmics rejection , in pixels. (used only if mode=0). Session keyword: RADIUS; default: 2 |

**Note:** a) It is also possible to remove cosmic rays beforehand with a median filter working only along the y-axis (use rady = 0 in command FILTER/MEDIAN).

b) If possible nightsky lines are not well aligned with the y-axis, mode 1 is to be preferred.

c) Cosmics rays rejection is only performed in mode=1.

**Examples:**
```
SKY/LONG mysky myobject 10,15,35,40 3 0
```

## TUTORIAL/LONG *stdred/long* PB 23-MAR-93

**Purpose:** The main commands of the package Long are executed in sequence to demonstrate a typical reduction session.

**Syntax:** `TUTORIAL/LONG`

**Note:** Test images are loaded from the area MID_TEST.

**Examples:**
```
TUTORIAL/LONG
```

## VERIFY/LONG *stdred/long* PB 01-APR-93

**Purpose:** Checks conformity of files in the long-slit context

**Syntax:** `VERIFY/LONG file mode`

| | |
|---|---|
| `file` | name of the file |
| `mode` | check mode. Possible values: IMA, TAB, OTIME, AIRMASS |

**See also:** None

**Note:** None

**Examples:** `VERIFY/LONG ccd0023 IMA`

## XIDENTIFY/LONG *stdred/long* PB 23-MAR-93

**Purpose:** Plots a row of the arc spectrum and invokes the graphical user interface XIdent for interactive lines identification.

**Syntax:** `XIDENT/LONG [wlc] [ystart] [lintab] [tol]`

| | |
|---|---|
| `wlc` | Wavelength calibration frame Session keyword: WLC; no default value |
| `ystart` | Starting row number (pixels) for identification and calibration. Session keyword: YSTART; default : 0 |
| `lintab` | Line table, as generated by command SEARCH/LONG Session keyword: LINTAB; default: line.tbl |
| `tol` | Tolerance for the identification of lines, in world coordinates. This parameter must be set if the interactive pointing generates too often the error message: "Feature not found, try again..." (No session keywords controls this parameter) Default: 4. |

**See also:** SEARCH/LONG, GRAPH/LONG, CALIBRATE/LONG, CUTS, SET/GRAPH, IDENTIFY/LONG

**Note:** 1) This command requires the preliminary creation of a line table with SEARCH/LONG.

2) If YSTART=0, its value is automatically set to the central row number.

3) The arc spectrum is plotted with respect to the current set-up of plotting parameters. To adjust the plotting, use commands CUTS or SET/GRAPH.

**Examples:** `IDENT/LONG ystart=200`

# Appendix L

# Standard Reduction - optopus

# Standard Reduction Commands

# Context: optopus

# CREATE/OPTOPUS

## CREATE/OPTOPUS *stdred/optopus* 10-SEP-1991 AG

**Purpose:** Create the input table for HOLES/OPTOPUS command, from ASCII file. Epoch of coordinates is written in descriptors of created table.

**Syntax:** `CREATE/OPTOPUS inp_file [out_tab] [fmt_file] [old_equinox]`

| | |
|---|---|
| `inp_file` | input ASCII file. |
| `[out_tab]` | output table. Default name is "opto1.tbl". |
| `[fmt_file]` | format file to by modified by the user at the start of the Optopus run accordingly to his/her own input ASCII file. In case gives as input "copy" the command will copy the standard format file in the directory. However, if will not copy this format file if a file with the same name is already present in the working directory. Default value is "copy". |
| `[old_equi]` | epoch of RA and DEC coordinates in input file. This value is written in the descriptor TABEQUI of the output table. Default is 1950.0. The input format of this parameter must be YEAR.yyyyy. |

**Note:** A format file named "opto.fmt" is present in the directory
$MIDASHOME/$MIDVERS/stdred/optopus/incl (UNIX) or
MID_DISK:['MIDASHOME'.MIDVERS'.stdred.optopus.incl] (VAX/VMS),
and can be copied and subsequently modified in the user's own directory.

**Examples:**

```
CREATE/OPTOPUS myfile.dat

CREATE/OPTOPUS myfile.dat myoptotab

CREATE/OPTOPUS myfile.dat myoptotab ?  1985.0

CREATE/OPTOPUS myfile.dat myoptotab myfor.fmt 1975.76776

CREATE/OPTOPUS myfile.dat ?  ?  1991.767765894
```

DRILL/OPTOPUS *stdred/optopus* 03-SEP-1991 PJG

**Purpose:** Write OPTOPUS drill command file in ASCII format using positions stored in a table.

**Syntax:** `OPTOPUS/DRILL in_table [name]`

| | |
|---|---|
| `in_table` | name of table file with OPTOPUS star plate positions |
| `name` | name of ASCII output file with drill commands for drilling machine. Default is the table name with the '.dat' extension. |

**Note:** The input table file must have x,y,z drill positions in columns with labels of X, Y, and Z, respectively. The type of each object must be given in column TYPE.

**Examples:** `DRILL/OPTO sa94`
create a OPTOPUS drill file from the table 'sa94.tbl' with the default name 'sa94.dat'.

`DRILL/OPTOPUS field193 drill21`
read drill positions from table 'field193' and produce the command file 'drill21.dat'.

# HOLES/OPTOPUS

## HOLES/OPTOPUS *stdred/optopus* 13-SEP-1991 AG

**Purpose:** Calculate X and Y positions of guidestars and scientific targets on Optopus plate from RA and DEC equatorial coordinates using the center provided by the user or calculating it automatically upon request. Checks are made to point out objects falling out of the plate limits or in the so called "forbidden area". Overlaps between big guidestars, big and small guidestars, big guidestars and objects, small guidestars and objects and objects and objects are also notified to the user.

**Syntax:** `HOLES/OPTOPUS [inp_tab] [out_tab] [HH,MM,SS.sss]`

`[+/-DD,AM,AS.ss] [ac_flag] [p_flag] [old_eq,new_eq]`

`[inp_tab]` table created by the command CREATE/OPTOPUS and containing columns :IDENT, :TYPE, :CHECK, :OLDRA, :OLDDEC, (if the command PRECESS/OPTOPUS has been used), :RA and :DEC. Default name is "opto1.tbl".

`[out_tab]` output table, with :X, :Y, and :Z columns, instead of :RA and :DEC. Default name is "opto2.tbl".

`[HH,MM,SS.sss]`
Right Ascension of center of the plate. If defaulted, it is automatically calculated by taking the mean (see STATISTIC/TABLE command) of column :RA in input table or by reading the value set with the command SET/OPTOPUS CRA=.... (see documentation to SET/OPTOPUS).

`[+/-DD,AM,AS.ss]`
Declination of center of the plate. If defaulted, it is automatically calculated by taking the mean (see STATISTIC/TABLE command) of column :DEC in input table or by reading the value set with the command SET/OPTO CDEC=... (see documentation to SET/OPTOPUS).

`[ac_flag]` Character flag which enables or desables the automatic calculation of the center. Default value is "Y" for yes. If [ac_flag] is set to "Y" and RA and DEC of the center are defaulted, then the center is determined by averaging :RA and :DEC columns of input table. If, on the contrary, [ac_flag] is set to "N" and RA and DEC of the center are defaulted, the corresponding values are read from the keywords set with SET/OPTO CRA= CDEC= command (see documentation).

`[p_flag]` Character flag which enables or desables the automatic precession of the center. Default value is "Y" for yes. [p_flag] should be set to "N" if the user has already precessed the center coordinates to [new_equinox]. In case of an automatic determination of the center from an input table already precessed by the command PRECESS/OPTOPUS, [p_flag] is automatically set to "N".

`[old_eq,new_eq]`
Old and new epoch (before and after precession) of center coordinates. Default values are 1950.0 for [old_eq] and 2000.0 (or the epoch calculated from year,month,day of the observation if these values have already been set by the command SET/OPTOPUS DATE=... (see documentation)) for [new_eq].

**Note:** Any parameter of HOLES/OPTOPUS command can be defaulted. It is actually possible to default all of them having previously used the SET/OPTOPUS command (see documentation) to select convenient values which will be stored in keywords initialized in Optopus context. It is convenient to flip from the command HOLES/OPTOPUS to the command MODIFY/OPTOPUS and viceversa to check that all the objects in "wrong" position have been actually deleted from HOLES/OPTOPUS input table. One can be sure this has happened when no warning messages are displayed after having issued the command HOLES/OPTOPUS. (See also documentation to MODIFY/OPTOPUS command).

**Examples:** `HOLES/OPTOPUS op1 op2 00,35,43.345 -00,11,53.56 n y 1950,1991.76`

`HOLES/OPTOPUS op1 op2 ? ? y`

`HOLES/OPTOPUS ? ? 00,35,43.345 -00,11,53.56 n y 1950,1991.76776`

`HOLES/OPTOPUS op1 op2 00,35,43.345 -00,11,53.56 Y n`

`HOLES/OPTOPUS op1 op2`

---

## MODIFY/OPTOPUS *stdred/optopus* 12-SEP-1991 AG

**Purpose:** Plot on graphic screen positions of holes on Optopus plate so that plate limits, different types of objects (big and small guidestars and scientific targets) and objects in a (for any reason) "unconvenient" position are clearly shown. These you can select by cursor and in case reject by typing a "D" or a "d" when prompted. It may happen that two or more objects are too much close for a correct identification. In this case, the use of the command ZOOM/OPTOPUS (see documentation) is strongly advisable.

**Syntax:** `MODIFY/OPTOPUS [table]`

`[table]` input table of HOLES/OPTOPUS command which contains :RA and :DEC columns. This is the basic table of any Optopus run and has been created by the command CREATE/OPTOPUS. Default name is "opto1.tbl".

**Note:** RA and DEC are spherical coordinates, but they have to be plotted on a plane: this implies that getting closer to the pole the circle representing the plate border on the plot will become an ellips of increasing eccentricity.

**Examples:** `MODIFY/OPTOPUS myoptotab`

`MODIFY/OPTOPUS`

# PLOT/OPTOPUS

## PLOT/OPTOPUS *stdred/optopus* 13-SEP-1991 AG

**Purpose:** Plot positions of holes on Optopus plate in X and Y coordinates using different symbols for different types of objects (big and small guidestars and scientific targets). Each object is identified by a sequential number. The limits of the so called "forbidden area" are also shown.

**Syntax:** `PLOT/OPTOPUS [table] [label] [EW_flip_flag]`

`[table]` output table of REFRACTION/OPTOPUS command which contains :X and :Y columns. Default name is "opto3.tbl".

`[label]` label that will be used to identify the plot. Default is "Optopus plate".

`[EW_flip_f]` Character flag to enable or desable the EAST-WEST flip of the plot, which may be useful in some cases. Default is "N" for no.

**Note:** Before using this command it is necessary to assign the plotter one wants to use with the command: ASSIGN/GRAPH plottername NOSPOOL and then, after successful completion of PLOT/OPTOPUS, the plot is sent to the plotter by: COPY/GRAPH plottername axes.plt (lowercase if on UNIX machine!).

**Examples:**

```
PLOT/OPTOPUS opto3 myfield y

PLOT/OPTOPUS opto3 f287

PLOT/OPTOPUS ? sa94 Y
```

PRECESS/OPTOPUS *stdred/optopus* 10-SEP-1991 AG

**Purpose:** Precess RA and DEC coordinates in table created by CREATE/OPTOPUS command to [new_equinox]. Precessed coordinates are put in columns :RA and :DEC, while the old ones are respectively in :OLDRA and :OLDDEC. Epoch of coordinates is updated in input table descriptors.

**Syntax:** `PRECESS/OPTOPUS [inp_tab] [new_equinox]`

`[inp_tab]` table created by CREATE/OPTOPUS command. Default name is "opto1.tbl".

`[new_equi]` New epoch of RA and DEC coordinates after precession. Descriptor TABEQUI (see CREATE/ OPTOPUS command) is updated. Default is 2000.0 or epoch calculated from year,month, day of the observation if these data have already been set by the SET/OPTO DATE=... command (see description).

**Note:** [new_equi] must be given in the fomat YEAR.yyyyy so, before using this command it is convenient to set the date of the observation by the SET/OPTOPUS command and then to use the SHOW/OPTOPUS command to make sure all the parameters have been correctly set: a value of the epoch corresponding to date of the observation will be automatically displayed.

**Examples:**

```
PRECESS/OPTOPUS

PRECESS/OPTOPUS myoptotab

PRECESS/OPTOPUS myoptotab 1991.76776

PRECESS/OPTOPUS ? 1991.76776876345
```

# REFRACTION/OPTOPUS

## REFRACTION/OPTOPUS *stdred/optopus* 16-SEP-1991 AG

**Purpose:** Correct for atmospheric refraction effects X and Y coordinates of objects on Optopus plate. For given plate-center coordinates (they must the same used by the command HOLES/OPTOPUS), specified observation time slot and wavelength range of interest, REFRACTION/OPTOPUS determines: (a) an optimal differential correction vector, (b) an optimal chromatic correction vector for the guidestars. In the end, all the relevant parameters determined by REFRACTION/OPTOPUS are written into descriptors of the output table, so that they can be read by READ/DESCRIPTORS or printed by PRINT/DESCRIPTORS commands.

**Syntax:** `REFRACTION/OPTOPUS [inp_tab] [out_tab] [year,month,day] [exp]`

`[lambda1,lambda2] [start_st_slot,end_st_slot] [opt_st] [ast_flag]`

`[inp_tab]` table created by the command HOLES/OPTOPUS, which contains X and Y coordinates of positions of holes on Optopus plate. Default name is "opto2.tbl".

`[out_tab]` output table, with corrected X and Y coordinates and column :NUMBER, which sequentially identifies holes on starplate. Its descriptors contain all relevant parameters created or updated by REFRACTION/OPTOPUS. Default name is "opto3.tbl".

`[year,month,day]`
date of the observation. The permitted input formats are:
(1) YEAR,MONTH(number),DAY or
(2) YEAR.yyyyy,0,0
Default value is "1999.,12.,31.".

`[exp]` Exposure time in minutes of Optopus plate. Default value is "0.".

`[lambda1,lambda2]`
Wavelength range to optimize the corrections for atmospheric refraction. Input format must be: LAMBDA1,LAMBDA2 both in Angstroem. Default value is: "3800,8000".

`[st_st_sl,en_st_sl]`
Sidereal time slot, that is sidereal time interval during which a given Optopus plate will probably be used. Input format must be: ST1.ss,ST2.ss. Default value is: "00.00,00.00".

`[opt_st]` Optimal sidereal time for correction determinations, that is sidereal time for which the corrections for atmospheric refraction are calculated. Input format must be: ST.ss. Default value is: "00.00".

`[ast_flag]` Y or y (for yes) and N or n (for no) are the alternatives. An [ast_flag] set to Y enables the automatic determination of the optimal sidereal time for correction determinations (see parameter [opt_st]). An input from the user is otherwise requested. Default value is "Y".

**Note:** Any parameter of REFRACTION/OPTOPUS command can be defaulted. It is actually possible to default all of them having previously used the SET/OPTOPUS command (see documentation) to select convenient values which will be stored in keywords initialized in Optopus context.
To correctly determine the sidereal time slot, it is useful to have a test run of the command REFRACTION/OPTOPUS before actually using it. This way one gets the sidereal times of the beginning and end of the night, that is its duration, which can be divided by 4 or 5, depending on how many Optopus plates one wants to use, to get the 4 (or 5) sidereal time slots (take care to allow for some 15-20 minutes of "warming up" at the beginning of the night, of course).
It is a good idea always to save all the parameters used in an Optopus session by writing them into the descriptors of the last created table, typically the output table of REFRACTION/OPTOPUS, with the command SAVE/OPTOPUS tablename, (see documentation). It will be possible later to retrieve them by using RESTORE/OPTOPUS tablename (see documentation).
We consider useful to remind here that it is possible to have a print-out of the descriptors of any table by using the commands:
(1) ASSIGN/PRINTER FILE filename
(2) PRINT/DESCRIPTORS table *
The descriptors values will be in the ASCII file "filename".

**Examples:**
```
REFRACTION/OPTOPUS op2 op3

REFRACTION/OPTOPUS op2 op3 1991,10,9 60 4000,8000 20,22 21 n

REFRACTION/OPTOPUS op2 op3 1991,10,9 60 4000,7500 20,22 ?  y

REFRACTION/OPTOPUS op2 op3 1991,10,9 60 ?  20,22 ?  Y
```

---

## RESTORE/OPTOPUS *stdred/optopus* 10-SEP-1991 AG

**Purpose:** Restore parameters used in old run of Optopus package by reading them from descriptors of table they had been saved into by SAVE/OPTOPUS command. The parameters are then written in keywords initialized in Optopus context.

**Syntax:** `RESTORE/OPTOPUS table`

`table` name of table previously used to save Optopus parameters with SAVE/OPTOPUS command.

**Note:** This command may be useful whenever you want to resume an old Optopus session, on a particular plate, without starting from the beginning, that is from the CREATE/OPTOPUS command. Let's suppose, for example you already have all the files, and just want to plot the positions of the holes on a starplate, without having to initialize the parameters all over again. In this case it's enough to give the command RESTORE/OPTOPUS table (if you were careful enough to use SAVE/OPTOPUS table in the end of the old Optopus session, of course), and then PLOT/OPTOPUS myXYtable.

**Examples:**
```
RESTORE/OPTOPUS myoptotab

RESTORE/OPTOPUS mytable
```

# SAVE/OPTOPUS

## SAVE/OPTOPUS *stdred/optopus* 10-SEP-1991 AG

**Purpose:** Save parameters currently set in Optopus context by writing them into descriptors of a table chosen by the user. These parameters can subsequently be retrieved by using the command RESTORE/OPTOPUS.

**Syntax:** `SAVE/OPTOPUS table`

`table` name of table whose descriptors will be used to store current parameters of Optopus context.

**Examples:** `SAVE/OPTOPUS myoptotab`

`SAVE/OPTOPUS mytable`

SET/OPTOPUS *stdred/optopus* 16-SEP-1991 AG

**Purpose:** Set parameters in Optopus context. Any parameter used by an Optopus context command can be set either by entering it in the command line or by using the command SET/OPTOPUS. To see the current settings of the parameters use the command SHOW/OPTOPUS (see documentation).

**Syntax:** `SET/OPTOPUS option1[=value1] option2[=value2] ...`

| | |
|---|---|
| `option` | Optopus parameters to be set: value = value of the option. The following options with their values are possible: |
| `DEFAULT` | no value; sets Optopus package default mode (see below). |
| `CRA` | Right Ascension of the center of the Optopus plate. Input format must be: HH,MM,SS.sss<br>Default value is "00,00,00.000". |
| `CDEC` | Declination of the center of the Optopus plate. Input format must be: +/- DD,AM,AS.ss<br>Default value is "+00,00,00.00". |
| `OLDEQ` | Old equinox of equatorial coordinates. Input format must be: YEAR.yyyyy<br>Default value is "1950.0". |
| `NEWEQ` | New equinox of equatorial coordinates. Input format must be: YEAR.yyyyy<br>Default value is "2000.0". |
| `LABEL` | Character string that will be used to identify all the plots in an Optopus package run, both on graphics screen and on printer. Default value is "Optopus plate". |
| `DATE` | Year,month and day of the observation. The permitted input formats are:<br>(1) YEAR,MONTH(number),DAY or<br>(2) YEAR.yyyyy,0,0<br>Default value is "1999.,12.,31.". |
| `EXTIM` | Exposure time in minutes of Optopus plate. Default value is "0.". |
| `WRANGE` | Wavelength range to optimize the corrections for atmospheric refraction. Input format must be: LAMBDA1,LAMBDA2<br>both in Angstroem. Default value is: "3800,8000". |
| `SITSLT` | Sidereal time slot, that is sidereal time interval during which a given Optopus plate will probably be used. Input format must be: ST1.ss,ST2.ss. Default value is: "00.00,00.00". OST = Optimal sidereal time for correction determinations, that is sidereal time for which the corrections for atmospheric refraction are calculated. Input format must be: ST.ss. Default value is: "00.00". |
| `ACFLAG` | Y or y (for yes) and N or n (for no) are the alternatives. An ACFLAG set to Y enables the automatic determination of the plate center in the command HOLES/OPTOPUS. An input from the user is otherwise requested. Default value is "Y". |
| `PFLAG` | Y or y (for yes) and N or n (for no) are the alternatives. A PFLAG set to Y enables the automatic precession of the plate center in the command HOLES/OPTOPUS. The center coordinates are otherwise not precessed. Default value is "Y". |
| `ASTFLAG` | Y or y (for yes) and N or n (for no) are the alternatives. An ASTFLAG set to Y enables the automatic determination of the optimal sidereal time for correction determinations in the command REFRACTION/OPTOPUS (see option OST). An input from the user is otherwise requested. Default value is "Y". |

# SHOW/OPTOPUS

`EWFLAG` Y or y (for yes) and N or n (for no) are the alternatives. An EWFLAG set to Y enables the EAST-WEST flipping of the plots produced by the command PLOT/OPTOPUS (see documentation). Plots are otherwise not flipped. Default value "N".

**Note:** Input to the options OLDEQ and NEWEQ is possible only in the format YEAR.yyyyy anyway, it may be useful to notice that every time one sets the date of the observation with SET/OPTO DATE=... and then one uses SHOW/OPTOPUS to see the currently set values of all the Optopus parameters, the value of the EPOCH (that is the EQUINOX) of the date is also displayed, so that it may be copied into NEW EQUINOX by using SET/OPTO NEWEQ=... .

**Examples:**

```
SET/OPTOPUS CRA=00,35,42.567 CDEC=-00,24,36.01

SET/OPTOPUS OLDEQ=1950.0 NEWEQ=1991.76776

SET/OPTOPUS WRANGE=4000,8000 EXTIM=120

SET/OPTOPUS DATE=1991,10,9

SET/OPTOPUS ACFLAG=Y ASTFLAG=n

SET/OPTOPUS SITSLT=20.4,22.5
```

---

## SHOW/OPTOPUS *stdred/optopus* 12-SEP-1991 AG

**Purpose:** Show the setup parameters for the Optopus package.

**Syntax:** `SHOW/OPTOPUS`

**Note:** For the meaning of the parameters consult the documentation of the SET/OPTOPUS command.

**Examples:** `None`

ZOOM/OPTOPUS *stdred/optopus* 12-SEP-1991 AG

**Purpose:** Blow up the area of the Optopus plate where objects overlap allowing an easier identification. The center of the area is chosen interactively with the cursor from the plot displayed by the MODIFY/OPTOPUS command. After the zooming, a cursor again enables the user to select objects in wrong position so that they can be rejected by typing a "D" letter (uppercase or lowercase) when a prompt appears. (See also documentation to MODIFY/OPTOPUS command).

**Syntax:** `ZOOM/OPTOPUS [table] [zooming_factor]`

`[table]` input table of HOLES/OPTOPUS command which contains :RA and :DEC columns and refers to the drawing of the Optopus plate plotted by the MODIFY/OPTOPUS command. Default name is "opto1.tbl".

`[zoom_fact]` Factor used to enlarge the chosen area of the plate. Default is 5.

**Note:** If after the first zoom one realizes it is not enough or too much, it is possible to use again the command ZOOM/OPTOPUS only changing [zoom_fact] and without having to issue again MODIFY/OPTOPUS to re-plot the Optopus plate and subsequently choose the center.

**Examples:**

```
ZOOM/OPTOPUS myoptotab 5

ZOOM/OPTOPUS ? 10

ZOOM/OPTOPUS myoptotab 10
```

# Appendix M

# Standard Reduction - pisco

# Standard Reduction Commands

# Context: pisco

# REDUCE/PISCO

## REDUCE/PISCO *stdred/pisco* 24-SEP-1991 OS,MS

**Purpose:** Perform complete reduction of polarimetric data obtained at La Silla with PISCO

**Syntax:** `REDUCE/PISCO catalog table sky calibration [mode]`

| | |
|---|---|
| `catalog` | input catalog of data files to be reduced |
| `table` | output table |
| `sky` | input sky (or dark) measurement file |
| `calibration` | input calibration file |
| `mode` | 1, 2 or 3:<br>1: X and Y channel are reduced separately<br>2: X and Y channel are reduced together<br>3: X and Y channel are reduced separately as well as together |

**See also:** CREATE/TABLE, EXTRACT/LINE, COMPUTE/IMAGE, FFT/...
MIDAS User Manual - Volume B

**Note:** The data format must conform to the one described in the 1989 March version of ESO Operating Manual No. 13 (PISCO). The format of old PISCO data may have to be adapted.
A description of the command and the format of measurement files and output table is given in Appendix F of Volume B of the MIDAS User Manual.
All data frames, including sky measurements, must have the double precision descriptor O_TIME set with element 7 (exposure time) not being zero.

**Examples:** `REDUCE/PISCO incat outtab sky calib 3`
This would process all files withe entries in the image catalog 'incat.cat' taking the file 'sky.bdf' as sky measurement and calibrate with frame 'calib.bdf'. The X and Y channels will be reduced separately as well as together. Results are stored in table 'outtab.tbl'.

# Appendix N

# Standard Reduction - spec

# Standard Reduction Commands

# Context: spec

# CENTER/HISTOGRAM

| CENTER/HISTOGRAM | *stdred/spec* | 22-MAR-95 PB |
|---|---|---|

**Purpose:** Median estimate and scale estimates of an image

**Syntax:** `CENTER/HISTOGRAM image`

| | |
|---|---|
| `image` | Name of the image |

**See also:** @a histogram, CUMULATE/HISTOGRAM

**Note:** The median of the image is estimated. The median of squared deviations to the median constitutes a robust scale estimate. Both values are displayed and stored OUTPUTR/R/1/2 for further usage in applications.

**Examples:** `CENTER/HISTOGRAM ccd0026`

COMPUTE/PARAL *stdred/spec* 22-MAR-95 AS

**Purpose:** Computes the parallactic angle and the atmospheric differential refraction. cf. Filippenko, A.V.: 1982, PASP, 94, 715. The command provides:

a) altitude in degrees ( 0 at horizon, 90 at zenith)

b) azimuth in degrees (south=0,west=90,north=180,east=270)
c) parallactic angle in degrees (-180 to 180)
d) differential refraction angle in arcseconds (negative values mean that the working wavelength image is at an altitude larger than reference wavelength image)

**Syntax:** `COMPUTE/PARAL ra dec st wave refw`

| | |
|---|---|
| `ra` | Right Ascension in format hour,min,seconds (seconds can be fractionary) |
| `dec` | Declination in format degree,min,seconds (seconds can be fractionary) |
| `st` | Sidereal time in format degree,min,seconds (seconds can be fractionary) |
| `wave` | working wavelength in Angstroms (default: 5000) |
| `refw` | reference wavelength in Angstroms (default: 5000) |

**See also:** REFRACTION/LONG

**Note:** 1) RA and Dec should be apparent coordinates.
2) The procedure assumes standard conditions for the refraction angle calculation: Alt = 2km; phi = +/- 30deg; P = 600 mm Hg; T = 7deg C; f = 8 mm Hg.
(not much dependence on f or T)
3) The procedure assumes the latitude of La Silla observatory (Schmidt telescope: -29deg15'25.8")

**Examples:** `COMPUTE/PARAL 11,04,0 -18,05,0 11,0,0 3500. 6000.`

Will give the following results:
Altitude : 78.62 degrees
Azimuth : 184.83 degrees
Parallactic Angle : -175.57 degrees
Differential Refraction Angle : -0.31 arcseconds

# CONTINUUM/SPEC

## CONTINUUM/SPEC *stdred/spec* 08-AUG-95 PB

**Purpose:** This command allows to fit the continuum of a spectrum by smoothing splines. A preliminary selection of continuum regions is performed using a min/max filtering. Interactive definition of the continuum regions is also possible. A smoothing spline is fitted through the continuum regions.

**Syntax:** `CONTINUUM/SPEC in out [radius/meth] [type] [smooth] [degree]`

| | |
|---|---|
| `in` | Name of the input spectrum (1D or 2D) |
| `out` | Name of the output spectrum |
| `radius/meth` | This parameter can be given as a number, in which case it will be interpreted as the radius of the min/max filter. It is also possible to provide a method name (SELE or FIT), in which case the min/max filter will be skipped and the procedure will start with the interactive selection (SELECT) or with the fitting (FIT). Default value: 3 |
| `type` | Spectrum type (ABSORPTION, EMISSION, ALL, Default: ABS). The type EMISSION must always be given on-line for emission spectra, in particular if one uses the methods Select or Fit.<br>The type ALL is used for pure continuum spectra, like flat-fields. In that case the full spectrum will be retained for the fit.<br>For all types, some sections of the spectrum can be interactively removed with the interactive selection. |
| `smooth` | Smoothing factor for spline fitting. The default value (10000.) will usually provide a polynomial approximation. The spline can be constrained closer to the data by decreasing this value. Several iterations to determine the optimal value can be performed with the syntax:<br>CONTINUUM/SPEC in out FIT s=... |
| `degree` | Spline degree (Default: 3). |

**See also:** INTERPOLATE/TT, WRITE/IMAGE

**Note:** 1) The dispersion direction is exepected to be parallel to the rows of the spectrum. If the input spectrum is a two-dimensional frame, all rows will be averaged before fitting and the result fitted continuum will be growed to the size of the input image.
2) For Flat-Field normalisation, use the options Type=ALL and a small smoothing factor to enforce spline interpolation.
e.g. CONTIN/SPEC lnflat nff ? ALL 1. 3
3) After a preliminary filtering and selection of continuum points is performed, the user will be prompted for interactive correction. The message:
> Interactive Correction (y/n/v/q, default: yes):
will appear. Four answers are possible:

a) Yes (y): A graphic cursor will be used to enter the lower, then the upper limit of one spectral region to be rejected (because it contains spectral features).

b) No (n): No more correction wanted, the procedure will go to spline fitting.

c) Values (v): The lower and upper limits of one region to be rejected will be entered by values. The syntax is of the command WRITE/IMAGE is accepted (first pixel identified by <, last pixel by >, real number means world coordinates, sign @ indicates pixel numbers).

d) Quit (q): The procedure will abort here.

**Examples:**

```
CONTINUUM/SPEC a1ss ca1 12 4

CONTINUUM/SPEC a1ss ca1 select

CONTINUUM/SPEC a1ss ca1 fit EMI s=1200.  d=2
```

# CORRELATE/LINE

## CORRELATE/LINE *stdred/spec* 14-AUG-1992 PB

**Purpose:** Performs the cross-correlation between table columns. The elements on which the operation is performed can be selected by a reference column and a weighting column can be specified.

**Subject:** Wavelength calibration, spectroscopy.

**Syntax:** `CORRELATE/LINE table_1 table_2 [pixel] [cntr,tol,rg,st]`

`[pos,ref,wgt] [refmin,refmax] [outima]`

`table_1` name of the first table

`table_2` name of the second table

`pixel` pixel step. This value is a scaling factor for the following parameters tol,rg,st. Default value is pixel = 1.

`cntr,tol,rg,st`
center,tolerance,range,step.
Defaults: 0., 0.05, 5., 0.1
The cross-correlation is performed in the range (center - range*pixel) to (center + range*pixel). The shift is incremented by steps of step*pixel. The value tolerance*pixel is used to associate elements between the two tables (see Notes).

`pos,ref,wgt` Name of position, reference and weight columns.
Defaults: X,Y,+. It is assumed that the name of these columns is the same for table_1 and table_2.
The special symbol + can be used for the reference and weight column to specify that either no reference or no weight required.

`refmin,refmax` Minimum and maximum values in the reference column. Default: +. The special symbol + can be used to specify that all the reference column must be considered.

`outima` Name of output cross-correlation image. Default: middummx.bdf

**Note:** a) The reference column is used as an additional condition to associate elements between the two tables. Two elements at positions (pos1) and (pos2) contribute to increment the cross-correlation image at the value (shift) if:
- the absolute value of the algebric sum (pos1-pos2+shift) is smaller than tolerance*pixel.
- the reference values are identical.
The test on reference values is omitted if the special symbol plus (+) specifies the reference column.

b) If values refmin and refmax are specified, the correlation is performed only on the elements of the table which reference value is in the range [refmin,refmax]. Only one value refmin can be specified to indicate that the range is [refmin,refmin].

c) Any selection set in table_1 or table_2 will be respected and the relevant elements ignored.

d) The weight column can be used to modify the incrementation. A constant weight of value unity is used if the weight column is specified by the special symbol plus (+).

**Examples:** `CORRELATE/LINE line lined 0.25 ? X,ORDER,PEAK 8,13`

CUMULATE/HISTOGRAM *stdred/spec* 22-MAR-95 PB

**Purpose:** Transforms a histogram image into the cumulated histogram

**Syntax:** `CUMULATE/HISTOGRAM in out`

| | |
|---|---|
| `in` | Name of 1D input histogram image |
| `out` | Output histogram |

**See also:** @a histogram, PLOT/HISTOGRAM

**Note:** None

**Examples:** `CUMULATE/HISTOGRAM hh1 ch1`

# DEBLEND/LINE

## DEBLEND/LINE *stdred/spec* 01-DEC-94 PB

**Purpose:** This routine provides de-blending of spectral lines by means of a simultaneous fit of up to six Gaussian profiles to a selected spectral region.

**Syntax:** `DEBLEND/LINE infile [fitim] [fitpar] [method] [contin] [input] [intab]`

| | |
|---|---|
| `infile` | the name of a wavelength-calibrated, one- or two- dimensional spectrum. If the spectrum is two- dimensional, one of the axis must have NPIX = 1 |
| `fitim` | the name of the output file in which the fit image is to be stored (Default: mdtmp1.bdf) |
| `fitpar` | the name of the output table file where the fit parameters are stored (Default: mdfit.tbl) |
| `method` | 1 to 3 letters specifying the fit option to be used. (default ABG) available options: first letter: A: all fit parameters can be varied to get the best fit W: center wavelengths held fixed at input values, heights and widths of lines can vary F: widths of lines held fixed at given values, centers and heights of lines can vary S: all fit parameters can vary, but all lines are required to have the same width in the fit. second letter: B: non-weighted fit (ie. all points have equal weight) P: Poisson-noise weighted fit: weight(i) = 1/datavalue(i) third letter: G: fit with Gauss profiles L: fit with Lorentz profiles |
| `contin` | continuum option. Option available: P: continuum is defined by two points |
| `input` | C: all initial parameters input interactively using cursor (Default option). T: all initial parameters read from an input table B: (available for W and S fit options) For fit option = W, input wavelengths are read from input table, remaining parameters are input interactively using cursor. For fit option = S, initial guess at full width half max of lines is input as a number in P7, remaining parameters are input interactively. |
| `intab` | input table name (Default ttemp.tbl also used to hold input values for p6 = C); see also under B input option |

**See also:** INTEGRATE/LINE, CENTER/GAUSS, CREATE/GUI ALICE, CONTINUUM/SPEC

Note: DBLEND is basically a simple program for deblending of spectral lines. In a specified spectral region, starting guesses for the centers, maximum values (heights) and full width half maxima (widths) of the lines are input for up to six lines, and a simultaneous least-squares fit using the selected profile and a gradient search method is used to obtain a fit to the lines. The data points are continuum-subtracted before the fit is made. In all cases, the continuum is considered to be a straight line between two points. The data points can be specified to all have equal weight (ie non-weighted fit, the default) or to be Poisson weighted (w(i) = —y(i)—). The centers, maxima, sigmas (when Gaussian profiles are fit), full widths at half maxima, integrated area under each curve, and the associated estimated errors, of the fitted profiles are output in a table, and an image of what the fit looks like is produced and stored in an image. For all options, DBLEND ends with a plot of the input image overplotted with the image of the fit and the individual fitted profiles.
The complications in DBLEND arise through the many options for fitting methods and input of parameters. First I will describe the treatment of the continuum, then I will return to the fitting and line paramter input options.
The continuum is defined by either two or four points, depending on the "contin" option specified. If "contin" = P, two points must be given, and the continuum is considered to be a straight line passing through these two points. If "contin" = R, four points must be given. These points are considered to define two regions, each of which is a continuum region. From each region, a single continuum point is derived. Its X value is the average of the X values of the two endpoints, and its Y value is the average of all the data values between the two endpoints, calculated from the spectrum itself. The continuum is then considered to be a straight line passing through these two "average" continuum points.
The simplest but slowest way to use DBLEND is with the input = C option, where all information is input interactively. First the input spectrum is plotted to the screen. Then the user is asked to mark two points with the cursor, defining a region which is to be "zoomed in on". This limited region is then plotted, so that the line parameters can be entered with greater accuracy. Next the user is asked to use the cursor to mark two points defining the beginning and end of the spectral region to be used in the fit. Then the cursor is used again to mark the two or four continuum points, and finally the starting guesses for the line parameters must be marked. For each line, the left half maximum point is marked, then the center, then the right half maximum point. As many lines as desired can be marked, but only the first six will be used by DBLEND. Note that all these points are automatically stored in a table called TTEMP, which also has the correct format to be used as an input table when DBLEND is run with the input = T option. This table is treated as a disposable temporary table by DBLEND, but is not deleted at the end in case you want to save it. If you do not choose to save TTEMP, you will find that these files build up in your directory, and you must be sure to delete them now and then.
The fastest way to run DBLEND is with input = T. In this case it is assumed that "intab" contains the name of a MIDAS table file containing all the needed input information. The difficulty with this method is that the input table must be constructed ahead of time. The columns in the table should be, in this order: 1) the position in the spectrum of the data point (in world coordinates) (X-axis value when plotted), 2) the data value of the data point (Y-axies when plotted), and 3) the equivalent of (1) in pixels. This is exactly the format output by GET/GCURSOR (along with several additional columns), so the input table could be made by the user in a separate step using GET/GCURSOR. The points in the table, one per row, should be, in this order: 1) the beginning of the spectral region to be fit, 2) the end of the spectral region to be fit, 3) the first continuum point, 4) the second continuum point, [3a) the third continuum point (if contin = R), 4a) the fourth continuum point (if contin = R),] 5) the left half maximum point of the first line, 6) the center of the first line, 7) the right half maximum point of the first line; then 5) - 7) are repeated for all lines to be fit, up to six lines total. Note that if contin = R, the input table is changed slightly in the course of the fit, as the four continuum points are replaced by the two averaged continuum points derived from them.
The final input option, input = B, is only valid for the W and S fitting options. Its action is context-dependent and will be described below along with the two applicable options.
The most general of the fitting options is method = A. With this option, all the line parameters are allowed to vary freely to achieve the best fit.
The method = W option holds the positions of the line centers constant at the input values, while the heights and widths can vary to achieve the best fit. When the input = B option is used with this fitting option, the line centers to be used are assumed to be in the input table whose name is given in the intab parameter. The centers must be in a column labeled :X, and some arbitrary label must be given in a column labeled :IDENT. The table may contain other columns as well, since the columns to be used are identified by their labels, not their positions in the table. The positions

# DISPERSION/HOUGH

**Examples:** `DEBLEND/LINE spec4`
The input spectrum in file SPEC4.BDF will be treated. The default options method = A (all parameters free to vary), contin = P (two points define continuum), input = C (interactive input) will be used. The image of the fit will be stored in file MDFIT1.BDF, and the final fit paramters will be stored in file MDPAR.TBL.

`DEBLEND/LINE spec4 ? ? abl`
Same as above, but Lorentz curves will be fitted instead of Gauss curves.

`DEBLEND/LINE spec4 sp4fit sp4par f ? t lines`
The spectrum SPEC4 will again be treated. This time we have chosen to use method = F and input = T, with the input table being in the file LINES.TBL. The continuum option is again defaulted to P. The image of the fit will be stored in file SP4FIT.BDF, and the final fit paramters will be stored in file SP4PAR.TBL.

`DEBLEND/LINE spec4 sp4fit sp4par s ? b 2.1`
This example is similar to the second example, except that method = S has been chosen, where all the lines in the fit will have the same line width. All input will be given interactively, except for the input line width, for which the value 2.1 (in world coordinates) will be used.

---

## DISPERSION/HOUGH *stdred/spec* 22-MAR-95 PB

**Purpose:** This low-level command is used in the Long and Echelle packages to determine dispersion relations by Hough transform. A number of default keywords (AVDISP, WCENTER, LINTAB, LINCAT) are supposed to exist in the calling package.

**Syntax:** `DISP/HOUGH [wdisp] [wcent] fr_sp [line] [cat] [mode] [range] [vfl]`

| | |
|---|---|
| `wdisp` | Average dispersion, defaulted to session keyword AVDISP. |
| `wcent` | Central Wavelength, defaulted to session keyword WCENT. |
| `fr_sp` | frame specifications, as npix,start,step |
| `line` | Name of line table, defaulted to keyword LINTAB |
| `cat` | Name of line catalog, defaulted to keyword LINCAT |
| `mode` | Method (LINEAR,1D,3D,NOLINEAR). Default: LINEAR |
| `range` | Range of influence function. Default: 2.5 |
| `vflag` | Visualisation flag (Keep/NoKeep/Visualize)<br>Keep will store the HT on disk (in file middummh.bdf)<br>Visualize will display the HT. |

**Note:** For a description of the method see Ballester, 1994, Astron. Astrophys., 286, pp. 1011-1018.

**Examples:** `DISPERSION/HOUGH ? ? IMSIZE(1),1.,1.`

## EXTINCTION/SPECTRUM *stdred/spec* 13-JUN-1986 JDP

**Purpose:** Correct spectra for interstellar or atmopsheric extinction.

**Subject:** Spectroscopy, Calibration, Extinction.

**Syntax:** `EXTINCTION/SPECTRUM inframe outframe scale [table] [col]`

| | |
|---|---|
| `inframe` | the frame to be correct for extinction |
| `outframe` | the frame to hold the corrected data |
| `scale` | the scaling factor: a) E(B-V) if interstellar reddening b) AIRMASS if atmospheric extinction. Defaulted to the AIRMASS descriptors from the frame. |
| `table` | the table with the wavelength dependence of the reddening or the atmospheric extinction, defaulted to ATMOEXAN (cf. below) |
| `col` | the column of the table to be used (defaulted to: 2) |

**Output:** Outframe

**Note:** The table supported by MIDAS to correct for interstellar extinction is:
1a) INSTEXAN contains in column #1 the wavelength in Angstroem units and in columns 2-8 seven different extinction laws the references for which are as follows:
col. 2: Savage, B.D., Mathis, J.S., Ann. Rev. Astron. Astrophys. No. 17, 73 (1979): Galaxy
col. 3: Nandy et al., MNRAS Vol. 196, 955 (1981): LMC
col. 4: Prevot et al., Astron. Astrophys. Vol. 132, 389 (1984): SMC
col. 5: Koornneef, J., Code, A.D., ApJ Vol. 247, 860 (1981): LMC
col. 6: Seaton, M.J., MNRAS Vol. 187, 73P (1979): Galaxy fit
col. 7: Howarth, I.D., MNRAS Vol. 203, 301 (1983): Galaxy fit
col. 8: Howarth, I.D., MNRAS Vol. 203, 301 (1983): LMC fit
All laws are normalized to A/E(B-V) = 3.1 at 5550 A and interpolated and rebinned to constant step in wavelength (10 A). Note that the wavelength coverage is very different for the various data sets.
1b) The contents of INSTEXNM is the same except for use with spectra calibrated in nanometers
Correction for atmospheric extinction is possible with:
2a) ATMOEXAN containing the wavelength in Angstroem units in column #1 and the extinction law for La Silla as described by H. Tug in the ESO Messenger No.11, December 1977, and listed in the ESO User's Manual.
2b) ATMOSNM which is identical to ATMOEXAN except for wavelengths in nm
If "scale" is defaulted, descriptor "AIRMASS" is read from frame "inframe".
These tables are stored in MID_EXTINCTION:

**Examples:** `EXTINCTION/SPECTRUM OBSERVED CORRECTED`

# FILTER/RIPPLE

## FILTER/RIPPLE *stdred/spec* 25-NOV-1985 DB

**Purpose:** correct one-dimensional images for periodic ripple (e.g. in Reticon data)

**Subject:** Fourier transforms, periodic noise, microphonics.

**Syntax:** `FILTER/RIPPLE frame outframe period [start,end]`

| | |
|---|---|
| `frame` | name of input frame |
| `outframe` | name of output frame |
| `period` | ripple period (in pixels) |
| `start,end` | numbers of first and last pixel of range over which analysis is to be performed |

**Note:** The program internally recognizes near-integer periods and accordingly uses two different algorithms. For "odd" periods an internal rebinning is done (the result of which is not written on disk) to a step of 1/20 of the period and then back to unity. Both rebinnings do not include an interpolation as this would require knowledge of the nature and origin of the ripple.

"start" and "end" default to the first and last pixel of "frame". It is advisable to specify "start" and "end" so as not to include any spectral or instrumental features (remember that often the first and last few pixels of a spectrum contain rubbish!). The ripple correction will in any case be done on the entire frame.

The program works best if the slope of the spectrum (or the portion thereof which is analyzed) is small although an attempt is made to compensate its effect.

**Examples:** `FILTER/RIPPLE OLD NEW 8 200,700`
correct ripple in frame "OLD" with period 8, determine the amplitude over the pixels 200-700

`FILTER/RIPPLE CES RESULT 3.356`
correct frame "CES" for ripple with period 3.356, the entire frame is used for the amplitude diagnostics

**Restrictions:**
a) works in pixel space only with START = 1 and STEP = 1
b) if a significant portion of the pixels contain negative fluxes, the program may abort

## GRAPH/SPEC *stdred/spec* 22-MAR-95 PB

**Purpose:** If no graphic window is already created, creates a long graphic window adpated for spectroscopy

**Syntax:** `GRAPH/SPEC [size] [position] [id]`

| | |
|---|---|
| `size` | dim_x,dim_y. Size of the graph. Default: 800,350 |
| `position` | off_x, off_y. Position of the graph. Default: 1,1 |
| `id` | graph window ID. Default: 0 |

**See also:** CREATE/GRAPH, DELETE/GRAPH

**Note:** None

**Examples:** `GRAPH/SPEC`

## MERGE/SPECTRUM *stdred/spec* 17-AUG-1992 PB

**Purpose:** Merges two 1D spectra. The algorithm computes a weighted average in the overlapping region of adjacent orders. The normalized weight is a linear ramp between 0 and 1 in the overlapping region.

**Syntax:** `MERGE/SPECTRUM spec1 spec2 out [interval] [mode] [var1] [var2]`

| | |
|---|---|
| `spec1` | input image, sampled in the wavelength space. Corresponds to a extracted and rebinned spectrum. |
| `spec2` | similar to spec1. This frame can precede or follow spec1 in the wavelength space. |
| `out` | name of the resulting output spectrum. |
| `interval` | wavelength interval to be skipped at both edges of the overlapping region (in pixels). Default is 5. |
| `mode` | Mode of computing the weights in the overlaps. Possible modes are: CONSTANT(default), RAMP or OPTIMAL. The mode OPTIMAL requires the two additional variance images var1 and var2. |
| `var1, var2` | Variance images for the computation of optimal weights. |

**Note:** Out of the overlaps, the spectra are copied to the output spectrum. Within the overlap, the weights formula are the following:
Method CONSTANT:
out = (spec1 + spec2)/2.
Method RAMP:
out = spec1*(1-c) + spec2*c,
where c is a ramp varying from 0 to 1 in the overlap. Method OPTIMAL :
out = (spec1*var1**2 + spec2*var2**2)/(var1**2+var2**2)

**Examples:** `MERGE/SPECTRUM spe0016 spe0017 spe1617 2`

# NORMALIZE/SPECTRUM

## NORMALIZE/SPECTRUM *stdred/spec* 01-NOV-1986 DB

**Purpose:** Approximate continuum of 1-D spectra for later normalization.

**Subject:** Spectroscopy, flux calibration, spectral analysis.

**Syntax:** `NORMALIZE/SPECTRUM inframe outframe [mode] [table] [batch_flag]`

| | |
|---|---|
| `inframe` | name of input frame |
| `outframe` | name of frame to hold the fit |
| `mode` | GCURSOR, to start from scratch, input from graphics cursor<br>= ADD, to add new points to existing fit, input with cursor<br>= DELETE, to delete points from existing point (with cursor)<br>= TABLE, to take positions and bin widths from "table" and integrate in "inframe" over corresponding bins. Default: GCURSOR |
| `table` | intermediate working table with cursor positions defining wavelengths and bin widths. In mode TABLE the flux in "inframe" is integrated over bins provided with "table". "table" must contain one column labeled :X_AXIS (with the wavelengths) and another one labeled :BIN_WIDTH (with the bin widths). Default: TABLE |
| `batch_flag` | if equal to Y all plots will be suppressed. Default: N |

**Output:** Intermediate table FIT1D is used, with columns :X_AXIS and :Y_AXIS.

**Note:** All data is internally written to table FIT1D on which the actual spline is to be made. Mode GCURSOR is self-evident. Enter data with cursor and ENTER key. Exit: space bar. The wavelengths are taken as the center of the bins. For the integration, a rectangular "transmission curve" is assumed. The points determined are plotted as is the fit derived from them. Modes ADD and DELETE can be used to interactively edit table FIT1D until a satisfactory fit is achieved. To delete a point, reply "*" to the question asked on your session terminal. NOTE that only the latest fit made with this command can be edited!

Note that the actual normalization needs, then, to be done like COMPUTE/IMAGE normalized = inframe/outframe.

**Examples:** `NORMALIZE/SPECTRUM RAW FIT`
start a fit of image RAW, input is expected from graphics terminal, result to be written to new image FIT

`NORMALIZE/SPECTRUM RAW FIT D`
delete some ill fitting points (use graphics cursor), a new fit (new image FIT) will be made

`NORMALIZE/SPECTRUM SPECTRUM CONTINUUM T LAMBIN`
use wavelengths and bin width in table LAMBIN to accordingly integrate the flux in image SPECTRUM

## OVERPLOT/IDENTIFICATION *stdred/spec* 20-DEC-1986 RHW

**Purpose:** Overplot line identifications on the already displayed spectrum. The position on the x-axis and the corresponding identified lines are defined in two columns of the table.

**Syntax:** `OVERPLOT/IDENTIFICATION [table] [xpos] [ident] [ypos]`

| | |
|---|---|
| `table` | name of the table with the line identifications; default: LINE |
| `xpos` | reference to column with positions. The column must be of REAL*4 type; default :X |
| `ident` | reference to column with identifications. The column can be of any type. The display format of the column is used in the plotting; default :IDENT |
| `ypos` | code to define the vertical positions of labels as TOP (default) or BOTTOM |

**See also:** OVERPLOT/IDENT, SET/PLOT, PLOT

**Examples:**
```
OVERPLOT/IMAGE MYFRAME

OVERPLOT/IDENTIFICATION MYTABLE :X :LABEL
```

**Restrictions:**

A frame is assumed to have been plotted on the graphics display. Columns with positions are R*4 type.

# REFRACTION/LONG

## REFRACTION/LONG *stdred/spec* 01-DEC-94 AG

**Purpose:** Long slit observations suffer from differential atmos- pheric refraction even at moderate airmass conditions. Light losses for all wavelengths of your input image are calculated and added to the output spectra, which contains the intrinsic flux you would observe without any light losses due to diff. refraction and slit size. The parallactic angel of the slit you should (have) observe(d) your target with, is given to minimize light losses.

**Syntax:** `REFRACTION/LONG inim outim [mode]`

| | |
|---|---|
| `inim` | input image (one-dimensional) |
| `outim` | output image (one-dimensional) |
| `mode` | interaction mode (INTERACTIVE or AUTO) In mode INTERACTIVE, the user is prompted to provide the different parameters. In mode AUTO the values are read from keywords INPUTC, INPUTI, INPUTR, as indicated in the Note. |

**See also:** EXTINCTION/LONG, CALIBRATE/FLUX

**Note:** During calculations you are asked for:

1) The unit of the spectrum [INPUTI(1)] (1=Angstroem, 2=nm, 3=m, 4=other) and if the option other (4) was chosen: The exponent of the unit [INPUTI(2)]

2) Observing site (telescope) [OUT_B(1:9)] or coordinates a) latitude [degree (-90 to 90)] [INPUTR(2)] b) longitude [degree (-180 to 180)] [INPUTR(3)] c) altitude [meter] [INPUTR(4)] 3) Temperature [deg. Celsius] (at observing time) [INPUTR(5)] 4) Water contents of atmosphere [INPUTR(6)] (Moisture, see interactiv graphic) 5) Target coordinates a) right ascension [hh.mmss] [INPUTR(7)] b) declination [hh.mmss] [INPUTR(8)] 6) wavelength [Angstrom] (centered to the slit) [INPUTR(9)] 7) Seeing [arcsec] [INPUTR(10)] 8) astronomical angle [degree] [INPUTR(11)] (of slit you want to observe with) 9) Slit size a) length [arcsec] [INPUTR(12)] b) width [arcsec] [INPUTR(13)] 10) Observing date a) year [INPUTI(3)] b) month [INPUTI(4)] c) day [INPUTI(5)] 11) Observing time [U.T.] (start of exposure) [INPUTR(14)] 12) exposure time [min] [INPUTR(1)] (the time the program will use is : observing time + 1/2 exposure time)
References: 1) C.W. Allen, Astrophy. Quantities, 3rd edition p.118ff 2) De Ball, Lehrbuch der sph. Astronomie, Leipzig 1912 3) Duffett-Smith, Practical Astr. with your Calculater 4) Filippenko, A., P.A.S.P. (1982), 94, p.715

**Examples:** `REFRACTION/LONG spectrum spec2`

## REGRESSION/ROBUST *stdred/spec* 22-MAR-95 PB

**Purpose:** This commands performs a robust multi-variate regression by Least Median of Squares, using the PROGRESS algorithm (Rousseuw, 1987).

**Syntax:** `REGRESSION/ROBUST tab y x1[,x2,..,xn] [file] [out_col] [res_col]`

| | |
|---|---|
| `tab` | Table name |
| `y` | Column containing the dependent variable |
| `x1,x2..` | Columns containing the dependent variables. |
| `file` | Name of the ASCII file containing information on the different steps of the regression. (Default: progress.out). |
| `out_col` | Column containing the robustly estimated values This column will be created if it does not exist (Default :WCALC) |
| `out_col` | Column containing the residuals. This column will be created if it does not exist (Default :RESIDUAL). |

**See also:** REGRESS/POLY, REGRESS/LINEAR

**Note:** None, but for more information, read the book:
"Robust regression and outliers rejection",
Wiley and Sons, New-York, 1987.

**Examples:** `REGRESSION/ROBUST LINTAB :AUX :ORDER,:X,:X2 ?  :AUX`

# ROTATE/SPEC

## ROTATE/SPEC *stdred/spec* 14-JAN-1994 PB

**Purpose:** Rotate and optionally flip echelle images. MIDAS spectral packages expect that the wavelength increases from left to right and for echelle spectra that the order numbers increase from top to bottom. All possible combination of rotation and flip are made available by the parameters angle and flip_axis.
Original files can be removed after processing with the option mode=DELETE in order to save disk space.
The command works on catalogues containing the input images. The catalogue is created with:

CREATE/CATALOG catalogue dirspec (See HELP)

**Syntax:** `ROTATE/SPEC cat [root] [meth] [flip] [angle] [mode]`

| | |
|---|---|
| `cat` | name of the catalogue with the input images |
| `root` | root of the name of the resulting images (up to 4 alphanumeric characters starting with a letter). Default: rot |
| `meth` | Processing method, either ROTATE or FLIP.<br>ROTATE - rotation only. The angle of rotation is defined by parameter angle.<br>FLIP - rotate and flip, (default) Angle of rotation and flip axis are defined by angle and flip_axis. |
| `flip` | Flip Axis. Used only if meth=FLIP. Possible values: X,Y,XY Default: X. See command FLIP/IMAGE for more information |
| `angle` | Angle of rotation of the image, in degrees. Default: 90.0 degrees. |
| `mode` | Post-rotation mode, either KEEP or DELETE. Default value is KEEP: original files are conserved. Disk space can be saved with the option DELETE, which removes the input files after rotation. |

**Note:** none

**Examples:** `ROTATE/SPEC mycat mode=DELETE`

The images in mycat are rotated. You will find the output as rot0001, rot0002, ... The input files listed in the catalog mycat.cat are removed.

## SEARCH/LINE *stdred/spec* 18-DEC-1992 PB

**Purpose:** Search for emission/absorption lines in spectra.

**Subject:** Spectral analysis, wave calibration.

**Syntax:** `SEARCH/LINE frame w,t[,nscan] [table] [meth] [type]`

| | |
|---|---|
| `frame` | input spectrum |
| `w,t,nscan` | input parameters as:<br>w (window) is the approx. zero intensity line width in pixels.<br>t (threshold) is the absolute value of the background.<br>nscan number of scan lines of the input spectrum to be averaged before searching for the lines. Default is one. |
| `table` | output table. Default name is line.tbl The output table contains the columns:<br>:X - center of the line, in world coordinates<br>:Y - scan number for 2D spectra<br>:PEAK - value at the maximum/minimum |
| `method` | centering method as<br>GRAVITY - center of gravity of the 2 highest pixels with respect to the third one.<br>MAXIMUM - pixel with the maximum value<br>MINIMUM - pixel with the minimum value (default for absorption lines)<br>GAUSSIAN- center of the gaussian fitted to the line (Default method). |
| `type` | EMISSION (Default) or ABSORPTION |

**Note:** 1. Detection algorithm: Data in a running window is sorted to form a local histogram. An emission (absorption) feature is found when the central value in the window is greater than the median value plus (minus) the threshold. The width of the window corresponds to parameter w, the threshold to t.

2. Centering algorithm: Lines are centered according to the selected method:

- Gaussian: Centers a gaussian to the line, using the true pixel values (no local background involved).

- Gravity: Finds the center of gravity of two highest points of 3 points (the maximum and the two flanking pixels).

- Maximum: Finds position of the maximum.

3. The line positions in column :X are expressed in world coordinates. All spectra involved in the reduction session must in principle present the same start and step values. However, the package allows modifications of the world coordinates references (at your own risk).

**See also:** IDENTIFY/LINE, CALIBRATE/LINE, REBIN/WAVE

**Examples:** `SEARCH/LINE SPECTRUM 5,100 line GRAVITY EMISSION`
This command will search for emission lines in spectrum having more than five pixels and peak value over 100 in the local background.

# VERIFY/SPEC



**Purpose:** This low-level command checks the existence of calibration tables and if necessary updates the content of the keyword containing their name.

**Syntax:** `VERIFY/SPEC file dir keyw [type]`

| | |
|---|---|
| `file` | File name |
| `dir` | Environment variable containing the directory |
| `keyw` | Keyword name |
| `type` | File type (Default: TABLE) |

**Note:** None.

**Examples:** `VERIFY/SPEC LINCAT MID_ARC LINCAT`

# Appendix O

# Contributed commands - cloud

# Contributed Commands

# Context: cloud

# COMPUTE/ABSORPTION

## COMPUTE/ABSORPTION *contrib/cloud* 25-APR-1988 MP

**Purpose:** Induces absorption lines on a 1D spectrum The resulting frame is computed at a given instrumental resolution.

**Subject:** Spectroscopy, intersteller and intergalactic absorption, synthetic spectrum

**Syntax:** `COMPUTE/ABSORPTION in out [cm_table] [ad_table] [psf_frame]`

| | |
|---|---|
| `inframe` | input frame.To be created by the command COMPUTE/EMI |
| `outframe` | output synthetic spectrum |
| `cm_table` | table containing the cloud model: position, thermal width, column density, atomic transition for the lines to be created. A template can be found in MID_CLOUD:absc.tbl Default: absc |
| `ad_table` | table containing the atomic data. A template can be found in MID_CLOUD:absp.tbl Default: absp |
| `psf_frame` | image containing the instrumental response. To be created by command ABS/PSF. Default: psf |

**Note:** The redshift of the emission source and the option for the shape of the absorption profiles (maxwellian or voigtian) are to be entered via keywords CLDZ and CLDOP. For a complete description see MIDAS Users Guide: CLOUD CONTEXT

**Examples:** `COMPUTE/ABSORPTION backg result test`
the command will create two spectra, namely :
1- result1 or result0 : theoretical spectrum
2- result1c or result0c : theoretical spectrum convolved by the instumental PSF
the character 0 or 1 refering to the option for maxwellian or Voigtian absorption line profiles respectively.

## COMPUTE/EMISSION *contrib/cloud* 25-APR-1988 MP

**Purpose:** Creates a synthetic 1D spectrum consisting of a polynomial continuum and gaussian emission lines. This image is to be used by the command COMPUTE/ABSORPTION.

**Subject:** Spectroscopy, synthetic spectrum

**Syntax:** `COMPUTE/EMISSION outframe [em_table]`

| | |
|---|---|
| `outframe` | output synthetic spectrum |
| `em_table` | table containing the line parameters. A template can be found in MID_CLOUD:emi.tbl Default: indicates that no emission line will be created |

**Note:** The coefficients for the continuum and the image dimensions. are to be entered via keywords CLDCT and CLDDIM. For a complete description see MIDAS Users Guide: CLOUD CONTEXT.

**Examples:** `COMPUTE/EMISION backg line`

## CREATE/PSF *contrib/cloud* 25-APR-1988 MP

**Purpose:** Creates a 1D image of a normalized gaussian centered in 0. This image is to be used by the command COMPUTE/ABSORPTION.

**Subject:** Spectroscopy, point spread function

**Syntax:** `CREATE/PSF [outframe] fwhm`

| | |
|---|---|
| `outframe` | output image. Default: psf |
| `fwhm` | full width at half maximum of the gaussian in Angstroms |

**Note:** The stepsize of the ouput image is set by the keyword CLDDIM For a complete description see MIDAS Users Guide: CLOUD CONTEXT

**Examples:** `CREATE/PSF result 4.5`

# Appendix P

# Contributed commands - esolv

# Contributed Commands

# Context: esolv

# FROMOD/ESOLV

## FROMOD/ESOLV *contrib/esolv* JEH

**Purpose:** Frames from the ESO-LV archive are retrieved from optical disk, using the ESO-identifiers or selected entries in the input table, and converted to BDF-format.

**Syntax:** `FROMOD/ESOLV [mode] [colour] [intable column]`

| | |
|---|---|
| `mode` | Single character giving the mode in which the program should run: P(rompt), A(uto). In prompt mode (P) the user is prompted for the identifiers and hence p2, p3 and p4 are not used. Default 'P' |
| `colour` | Single character giving the colour of the image to be retrieved: 'B' for only blue frames, 'R' for only red frames, and 'A' for all frames; Default 'B'. |
| `intable` | With mode 'P' intable should be name of table with column (R*4) containing ESO-identifiers; Default |
| `column` | Label or column number of table with contains the identifiers; No default |

**Note:** The command takes care of the logistics of the frames being resident on 3 different disks. In a single run only frames from a single disk can be retrieved. The proper disk should have been inserted in the device. See the system management for information about the optical disks.
For more information see: The Surface Brightness Catalogue of the ESO-Uppsala Galaxies, A.Lauberts and E.A.Valentijn, ESO: 1989.

**Examples:**
```
Midas 001> FROMOD P B
"Give ESO-identifier: " 1180010 <cr>
"Give ESO-identifier: " 1180020 <cr>
"Give ESO-identifier: " <cr>
```
Retrieve the blue frames of 1180010 and 1180020 from optical disk.
```
Midas 001> SELECT/TABLE ESOLV :BTOT.lt.10.and.:T.eq.-5

Midas 002> FROMOD A A ESOLV :ESO_ID
```
Retrieve brightest ellipticals from optical disk in both colours, using selected ESO-identifiers from ESO-LV catalogue.

## MTABLV/ESOLV *contrib/esolv* 01-MAY-1991 JEH

**Purpose:** Find semidiameter of ellipses at given fraction of light

**Syntax:** `MTABLV/ESOLV [table] col1 col2 l_frac col3`

| | |
|---|---|
| `table` | to be completed |
| `col1` | to be completed |
| `col2` | to be completed |
| `l_frac` | to be completed |
| `col3` | to be completed |

**Note:** none

**Examples:** `to be completed`

## STATPL/ESOLV *contrib/esolv* 01-MAY-1991 JEH

**Purpose:** Computes mean and sd of table file column given in p2 and dumps that in table with name t'p2'

**Syntax:** `STATPL/ESOLV table col1 select disp`

| | |
|---|---|
| `in_table` | input table |
| `col` | column for which the statistics has to be done |
| `select` | select criterium for the column in p2 [all] |
| `disp` | display statistics flag: 0 = no display; 1 = display |

**Note:** Intended for statistics on ESOLV (PCAT) catalogue using Type = column #7 as reference

**Examples:** `still to come`

## TABFLV/ESOLV *contrib/esolv* 01-MAY-1991 JEH

**Purpose:** Lists the contents of special table file

**Syntax:** `TABFLV/ESOLV [table] ascii_file flag`

| | |
|---|---|
| `table` | to be completed |
| `ascii_file` | to be completed |
| `flag` | to be completed |

**Note:** none

**Examples:** `to be completed`

## TEXLV/ESOLV *contrib/esolv* JEH

**Purpose:** Prepare Tex file with selected parameters from ESOLV

**Syntax:** `TEXLV/ESOLV [table] tex_file`

| | |
|---|---|
| `table` | to be completed |
| `tex_file` | to be completed |

**Note:** none

**Examples:** `to be completed`

# Appendix Q

# Contributed commands - geotest

# Contributed Commands

# Context: geotest

# CREATE/ART_IMAGE

## CREATE/ART_IMAGE *contrib/geotest* 26-FEB-1992 PB

**Purpose:** Create a new artifical image frame.

**Subject:** Artificial frames, models, fractals.

**Syntax:** `CREATE/ART_IMAGE frame [dims] [starts,steps] [func_type] [coefs]`

`or`

`CREATE/ART_IMAGE frame = ref_frame [func_type] [coefs]`

| | |
|---|---|
| `frame` | name of new frame |
| `ref_frame` | reference frame, i.e. take all descriptors from that frame |
| `dims` | NAXIS,NPIX(1),...,NPIX(NAXIS); default: 2,512,512 |
| `starts,steps` | START(1),...,START(NAXIS),STEP(1),...,STEP(NAXIS)<br>default: = -1.3 for START(i), = 0.005 for STEP(i)<br>Start and step may have to be adapted depending on the chosen function and coefficients. |
| `func_type` | type of function used to define the content of the created image:<br>MANDEL for creation of Mandelbrot set<br>JULIA for creation of Julia set<br>LORENZ for creation of Lorenz attractors<br>ELFLY for simulation of a chaotic system (electronic fly)<br>COLWHEEL for creation of a colour wheel<br>Default: LORENZ |
| `coefs` | coefficients for the function which is used to create the image, number and meaning depends on the parameter func_type:<br>MANDEL zx,zy,uplimit default: .31,.04,100<br>JULIA zx,zy,uplimit default: .31,.04,100<br>LORENZ x0,y0,z0,h,a,b,n_iter<br>default: 1,1,1,.01,28,0.1,10000<br>ELFLY a,b,c,d,e,f,n_iter<br>default: 2.24,.43,-.65,-2.43,1.,300000 |

**Note:** For MANDEL and JULIA, the coefficients represent zx,zy,uplimit where zx and zy are the real and imaginary part of a complex number zz, which serves as the initial value for z in the formula:
z <- z*z + c
the real and imaginary parts of c are the x,y coords
If after 256 iterations z <= uplimit, the pixel at (x,y) is set to 0.0, else pixel <- iteration_no .
For ELFLY, the coefficients represent a,b,c,d,e,f,no_iter where a,b,c,d,e,f are control parameters and no_iter the number of loops for evaluation of the formulaes:
xx = sin(a*y) - z*cos(b*x)
yy = z*sin(c*x) - cos(d*y)
zz = e*sin(x)
x = xx, y = yy, z = zz.
the pixel value at resulting x,y coords (z is discarded) is increased by 1.0 .
For LORENZ, the coefficients represent x0,y0,z0,h,a,b,no_iter where x0,y0,z0 are the inital value for a point in space, h,a,b are coefficients and no_iter the number of loops for evaluating the formulae:
xx = x + 10.*h*(y-x)
yy = y = h*((-x*z)+a*x-y)
zz = z + h*(x*y-b*z)
x = xx, y = yy, z = zz.
the pixel value at resulting x,y coords (z is discarded) is set to 1.0 .

**Examples:**

```
CREATE/ART mandel ?  -1.6,-1.3,0.05,0.05 MANDEL

CREATE/ART julia P4=JULIA

CREATE/ART lorenz ?  -25,-25,0.1,0.1 LO .1,.1,.1,.01,28,2.66,4000

CREATE/ART elfly ?  -2.5,-2.5,0.01,0.01 ELFLY 2.24,.43,-.65,-2.43,1,300000
```

## CREATE/RAMP

*contrib/geotest* 07-JULY-1986 FM

**Purpose:** Generate uniform sloping background image, with mean flux per pixel of 100 units.

**Syntax:** `CREATE/RAMP image [slope] [angle] [dimension]`

| | |
|---|---|
| `image` | the output image name, |
| `slope` | slope of the plane (units per pixel side length; default = 1.0), |
| `angle` | integer position angle of the gradient (degrees; default = 0), |
| `dimension` | integer dimensions of axes of image (default = 128). |

**Note:** Position angle 0 gives a slope parallel to the x-axis, and angles up to 90 rotate the sloping plane in a counter-clockwise direction.

**Examples:**

```
CREATE/RAMP outim 1.5 60
```

# CREATE/SPC1

## CREATE/SPC1 *contrib/geotest* 07-SEPT-1986 FM

**Purpose:** Generate sinusoidal, sloping 1-dimensional image.

**Syntax:** `CREATE/SPC1 image [slope] [ampl] [period] [phase] [dim]`

| | |
|---|---|
| `image` | the output image name, |
| `slope` | the slope on which the sine wave is superimposed (default = 0.0); flux added to the periodic pattern in the I'th image pixel is slope*I, |
| `ampl` | half peak-to-peak (default = 10.0), |
| `period` | period of the sine wave (default = 8.5 pixel units), |
| `phase` | default = 0.0, |
| `dim` | ... of the output image (default = 128). |

**Note:** Normalization to a mean flux of 100 units per pixel is carried out.

**Examples:**
```
CREATE/SPC1 outim 1.5 12.0 8.9 0.5
```

## CREATE/SPC2 *contrib/geotest* 07-SEPT-1986 FM

**Purpose:** Generate a discrete 1-dimensional image with an arbitrary periodic pattern. For a period, p, p values are read in (or defaulted) to give the flux values of p successive pixels. Optionally, the periodic pattern can be superimposed on a constant slope.

**Syntax:** `CREATE/SPC2 image [period] [slope] [phase] [dimension]`

| | |
|---|---|
| `image` | the output image name, |
| `period` | integer value between 1 and 20 (default = 10), |
| `slope` | real slope of the image (default = 0.0); flux added to the periodic pattern in the I'th image pixel is slope*I, |
| `phase` | integer denoting the sequence number of the pattern element which goes into the first pixel of the image created (default = 1), |
| `dimension` | dimension of the output image (default = 128). |

**Note:** Keyword RIPVALS may be initialised prior to calling SPEC2, with p (period) integer values which will define the image; or, alternatively, if this is not done, use the default period (i.e. 10) and the default wave definition (i.e. 4,5,6,3,2,7,1, 4,5,0).

**Examples:**
```
WRITE/KEY RIPVALS/I/1/11 4,5,6,3,2,2,1,4,5,6,0

CREATE/SPC2 outim 11
```

## CREATE/SPC3 *contrib/geotest* 11-SEPT-1986 FM

**Purpose:** Generate an artificial spectrum with lines whose locations are defined by a MIDAS table. Either box or gaussian psfs are used (a delta function being used for the former if it is of sufficiently small width). Note that the input table must be created in one's area. The lines, defined in the table, are replicated three times with different centrings in each of the three juxtaposed "sectors", each of length 220 pixels. The first set of lines is centred on the left edge of the pixel; the second set on the pixel centre; and the third set is offset from the left edge of the pixel by the user-specified value.

**Syntax:** `CREATE/SPC3 image psf_option centring table boxwidth_or_fwhm`

| | |
|---|---|
| `image` | the output image name. |
| `psf_option` | a box of specified width (B, default); or a gaussian of specified full width half max. (G). |
| `centring` | positions of lines vis-a-vis pixel centres (in the third 220-pixel sector). Default is 0.645. |
| `table` | input table (single column, no specific label required), giving positions of lines. Default is LINES.TBL. |
| `boxwidth` | width of "box" PSF; if .LE. 0.001, then a delta function is used instead. (or) |
| `fwhm` | full width half maximum (greater than 0.2). Default of boxwidth or of fwhm is 1.0. |

**Note:** Input lines have normalized peak height. Depending on line centring and line width, the maxima of the actual lines after their mapping onto the pixel grid may be smaller.

**Examples:** `CREATE/SPC3 outim G 0.61 lines 1.75`

## CREATE/WAVE *contrib/geotest* 07-JULY-1986 FM

**Purpose:** Generate 2-dimensional sinusoidal background image, with mean flux per period of 0 units.

**Syntax:** `CREATE/WAVE image [amplitude] [period] [dimension]`

| | |
|---|---|
| `image` | the output image name, |
| `amplitude` | wave amplitude (half peak-to-peak; default = 10.0), |
| `period` | period of wave (default = 8.5), |
| `dimension` | integer dimensions of axes of image (default = 128). |

**Note:** none

**Examples:** `CREATE/WAVE outim 12.0 8.9`

# Appendix R

# Contributed commands - invent

# Contributed Commands

# Context: invent

# ANALYSE/INVENTORY

## ANALYSE/INVENTORY *contrib/invent* 14-DEC-1988

**Purpose:** In the VERIFY mode, the ANALYSE command verifies the used table of objects, and calculates the image parameters. In the NOVER mode the verification is omitted.

**Syntax:** `ANALYSE/INVENTORY frame table1 [table2] [ver_par] [debug] [out_psf]`

| | |
|---|---|
| `frame` | input frame |
| `in_tab` | input table (output of SEARCH command or other MIDAS table format file with at least two columns containing world coordinates of objects with labels :X and :Y) |
| `out_tab` | optional output table |
| `ver_par` | VER for verification (default) NOVER for non-verification |
| `deb_mode` | debug mod: DEBUG or NODEBUG; optional (used with small frames in case of trouble); default is NODEBUG |
| `out_psf` | name of an image frame to hold point spread function extracted from the input image. Effective only when the keyword ANALITER and at least one of keywords FULLPSF or UNDRSMPL have positive values. Subpixeling is applied in accordance with value of keyword UNDRSMPL. |

**Note:** In the NOVERIFY mode, the verification process is omitted. The table of objects is produced independently of the Inventory procedure and the ANALYSE/INV command calculates several image parameters, which can be used as final results and/or as input for the CLASSIFY/INV command.
In the VER mode, the command verifies the used table of object, which is the output of the SEARCH/INV command. Many of the entries are deleted, some new ones are added. The object positions are improved and the ANALYSE/INV command calculates several image parameters, which can be used as final results and/or as input for the CLASSIFY command.
The output table will contain all columns as described in the user manual (Volume 2, Chapter 4). However, the last 3 columns used for the error estimates for the position and magnitude will only be filled when ANALYSE/INVENT is used with a two-dimensional psf (FULLPSF larger than 0) and more than one iteration (ITERATE greater than 0).
Before running the command, check the values of the keywords used by this command with SET/INV A(nalyse) [A(ll)]

**Examples:** `ANALYSE/INVENT IMAGE TABLEA TABLEB VER`
Analysis on frame IMAGE objects listed in table TABLEA. Some of the objects are deleted, some additional objects are added. TABLEA should contain columns named X and Y which are obligatory. The presence of columns BG and INT may save some CPU time. Output table TABLEB contains 22 parameters for each object including positions, magnitudes, and sizes.

`ANALYSE/INVENT SMALLIMAGE TABLEA TABLEB VER DEBUG`
Displays on terminal plenty of intermediate results to help understanding how the analysis is done.

`ANALYSE/INVENT SMALLIMAGE TABLEA ? VER DEBUG`
The same as before but output table is not created.

`ANALYSE/INVENT IMAGE TABLEA TABLEB NOVER`
Analyses on frame IMAGE objects listed in table TABLEA. Obtained parameters are listed in table TABLEB for the same set of objects. The objects coordinates are not changed. The input table should contain columns X and Y.

`ANALYSE/INVENT SMALLIMAGE TABLEA ? NOVER DEBUG`
Displays on terminal plenty of intermediate results to help understanding how the analysis is done.

## CLASSIFY/INVENT *contrib/invent* 14-DEC-1988 RHW+AK

**Purpose:** The CLASSIFY/INV command uses the output table produced by the ANALYSE/INV command for dividing the objects into stars,galaxies and spurious objects.

**Syntax:** `CLASSIFY/INVENT in_table`

| | |
|---|---|
| `in_table` | input table containing the image parameters and where will be stored the classification results in the column #21 labelled CLASS |

**Note:** Before running the CLASSIFY/INV command, check the values of the keywords used by this command with SET/INV C(lassify).

**Examples:** `CLASSIFY/INV MYTABLE`
classifies objects listed in table MYTABLE which itself is an output of command ANALYSE/INVENT. Resulting classification is coded - stars: 1, galaxies: 2, unclassified: 0 - in column #21 labelled CLASS.

## SEARCH/INVENTORY *contrib/invent* 14-DEC-1988 RHW+AK

**Purpose:** Search objects in an image frame and store the parameters describing the identified objects in a table

**Syntax:** `SEARCH/INVENT frame table`

| | |
|---|---|
| `frame` | input file name in which the search command will operate |
| `table` | output table name in which the parameters describing the found objects will be stored |

**Note:** Before running the SEARCH/INV command, check the values of the keywords used by this command with SET/INV S(earch) [A(ll)]

**Examples:** `SEARCH/INVENT MYFRAME MYTABLE`
Searches for objects in frame MYFRAME and stores results in table MYTABLE. There are six columns in an output table labelled: IDENT, X, Y, BG, INT, and AR. They contain: identification, x and y world coordinates, local background, average intensity of 9 central pixels, and approximate object size in pixels.

# SET/INVENTORY

## SET/INVENTORY *contrib/invent* 14-DEC-1988 RHW+AK

**Purpose:** Display and modify the values of the keywords used by Inventory

**Syntax:** `SET/INVENTORY par1 [par2]`

| | |
|---|---|
| `par1` | S for the keywords concerning the command SEARCH/INV A for the keywords concerning the command ANALYSE/INV C for the keywords concerning the command CLASSIFY/INV |
| `par2` | A if you want to display All the keywords used by one command and modify them . |

**Note:** none

**Examples:** `SET/INV S`
adjusts most important parameters used in command SEARCH

`SET/INV A A`
adjusts all parameters used in command ANALYSE

## SHOW/INVENTORY *contrib/invent* 14-DEC-1988 RHW+AK

**Purpose:** Display the values of the keywords used by the Inventory package

**Syntax:** `SHOW/INVENTORY par1 [par2]`

| | |
|---|---|
| `par1` | S for the keywords used by the command SEARCH/INV A for the keywords used by the command ANALYSE/INV C for the keywords used by the command CLASSIFY/INV |
| `par2` | A to display All the keywords used by one command |

**Note:** none

**Examples:** `SHOW/INV S`
Shows most important parameters used in command SEARCH

`SHOW/INV A A`
Shows all parameters used in command ANALYSE

# Appendix S

# Contributed commands - mva

# Contributed Commands

# Context: mva

# CLUSTER/TAB

## CLUSTER/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Hierarchical clustering.

**Syntax:** `CLUSTER/TAB input output [method]`

| | |
|---|---|
| `intable` | input table |
| `outable` | output table |
| `method` | MVAR (minimum variance - default) MNVR (minimum variance - for large number of columns) SLNK (single link) CLNK (complete link) ALNK (average link) WLNK (weighted average link) MEDN (median) CNTR (centroid) |

**Note:** none

**Examples:** `CLUS/TAB INTAB OUTAB`
produces class assignments in top 9 layers of hierarchy in OUTAB (together with input data), and dendrogram representation on screen.

## CMDS/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Principal Coordinates Analysis (Gower's), Classical (Torgerson's) Multidimensional Scaling.

**Syntax:** `CMDS/TAB input_table output_table ncols._output_table`

| | |
|---|---|
| `input_table` | Input table name. |
| `output_table` | Output table name. |
| `ncols._output_table` | Number of columns in output table (no. of principal coordinates) Defaults to 3. |

**Note:** none

**Examples:** `CMDS/TAB INTAB OUTAB`
takes symmetric table of distances in INTAB.

## CORRESPONDENCE/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Correspondence Analysis.

**Syntax:** `CORRESPONDENCE/TAB input output row/col_analysis ncolumn outable`

| | |
|---|---|
| `input` | table name. |
| `output` | table name. |
| `row/col_analysis` | Row or column analysis; default R. Note that all results are detd., but results are output according to choice. |
| `ncolumn` | Number of coulmns in output table (no. of principle components); default 3. |
| `outable` | Eigenvectors output table name; default no table. |

**Note:** none

**Examples:** `CORRES/TAB INTAB OUTAB`
Carries out a CA on all columns of INTAB, and producing output projections on 3 factors in OUTAB. A keyword is written in OUTAB containing eigenvalues.

## EDIST/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Produce a table of standardized distances.

**Syntax:** `EDIST/TAB input_table output_table`

| | |
|---|---|
| `input_table` | Input table |
| `output_table` | Output table |

**Note:** none

**Examples:** `EDIST/TAB INTAB OUTAB`
produces rows x rows distances (having first standardized).

## KNN/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** K-Nearest Neighbours Discriminant Analysis.

**Syntax:** `KNN/TAB training_table no._of_gp.1_members test_table no._of_NNs`

| | |
|---|---|
| `input training` | set table name. |
| `number` | first "number" rows of the input training set belong to group 1; the remainder belong to group 2. |
| `test` | set table name. Number of nearest neighbours (default: 3) |

**Note:** none

**Examples:** `KNN/TAB INTAB1 5 INTAB2`
... takes first 5 rows of INTAB1 as group 1, remaining rows as group 2, and determines best assignments for rows-points of INTAB2. Screen output, only, is produced.

# LDA/TAB

## LDA/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Fisher's Linear Discriminant Analysis.

**Syntax:** `LDA/TAB input_table output_table`

`input_table` input table

`output_table` output table

**Note:** Singularities, caused by linear dependancies, can be circumvented by first carrying out a PCA and using a small number of principal components as input to LDA

**Examples:** `LDA/TAB INTAB OUTAB`

## MDA/TAB *contrib/mva* 28-AUG-1986 FM

**Purpose:** Multiple Discriminant Analysis, Discriminant Factor Analysis, Canonical Discriminant Analysis.

**Syntax:** `MDA/TAB input_table output_table eigenvectors`

`input_table` input table name.

`output_table` output table name.

`eigenvectors` output table name (defaults to none).

**Note:** Singularities, caused by linear dependance, can be circumvented by first carrying out a PCA, and using a small number of principal components as input to LDA. Singularity problems may arise in matrix inversion, due to very small values: if you get an IERR=2 message, multiplying the input data by an arbitrary constant may painlessly get around the difficulty.

**Examples:** `MDA/TAB INTAB OUTAB`

## MST/TABLE *contrib/mva* OCT-1985 BP/JDP

**Purpose:** Create a minimal spanning tree from a table containing points positions.

**Syntax:** `MST/TABLE intable outtable grid_size`

`intable` input table (with two columns called :X and :Y)

`outtable` table with the structure : :X1 :Y1 :X2 :Y2 :DIST :POINT_NB1 :POINT_NB2 (one entry per link created)

`grid_size` use 50.

**Note:** none

**Examples:** `MST/TABLE MYTAB MST 50.`
Compute the minimal spanning tree from the table MYTAB giving table MST as result

## PARTITION/TAB

*contrib/mva* 16-APR-1986 FM

**Purpose:** Non-hierarchical clustering (partitioning).

**Syntax:** `PARTITION/TAB input output [no_of_class] [alg] [min_card] [s_value]`

| | |
|---|---|
| `intable` | input table |
| `outable` | output table |
| `no_of_class` | number of classes sought (default 3) |
| `alg` | 1 or 2 (min. distances criterion, or exchange method; default = 1) |
| `min_card` | min. acceptable no. of items per class (default = 1) |
| `s_value` | seed for random determination of initial classes (default = 37519) |

**Note:** none

**Examples:** `PARTITION/TAB INTAB OUTAB`
produces class assignments for three classes in the final column of OUTAB.

## PCA/TAB

*contrib/mva* 28-AUG-1986 FM

**Purpose:** Principal Components Analysis, Karhunen-Loeve expansion.

**Syntax:** `PCA/TAB in_tab out_tab option row/col_anal ncols_table eigenvectors`

| | |
|---|---|
| `in_tab` | Input table name. |
| `out_tab` | Output table name. |
| `option` | Analysis option: 3 = PCA of correlation matrix (default); 2 = PCA of covariance matrix; 1 = PCA of sums of squares and cross-products matrix. |
| `row/col_anal` | Row/column analysis (R - default - or C; note that all results are detd., but that output is produced for either rows or columns (cf. so-called "Q-" or "R-mode" factor analysis). |
| `ncols_table` | Number of columns in output table (no. of principal components). Defaults to 3. |
| `eigenvectors` | Eigenvectors output table name (defaults to none). |

**Note:** none

**Examples:** `PCA/TAB INTAB OUTAB`
carries out a PCA on all columns of INTAB, using standardization of rows (leading to PCA on correlations), and producing output projections on 3 principal axes in OUTAB. A keyword is written in OUTAB containing eigenvalues.

## PLOT/TREE

*contrib/mva* 01-APR-1987 FM

**Purpose:** Plot output created by minimal spanning tree (MST) algorithm

**Syntax:** `PLOT/TREE intable [col_ref]`

| | |
|---|---|
| `intable` | input table |
| `col_ref` | x1, y1, x2, y2, - labels or column numbers (defaulted to the first four columns) |

**Note:** none

**Examples:** `PLOT/TREE MYTAB`
plots graph defined in table MYTAB.

# Appendix T

# Contributed commands - pepsys

# Contributed Commands

# Context: pepsys

# CONVERT/PHOT

## CONVERT/PHOT *contrib/pepsys* 30-DEC-1992 ATY

**Purpose:** The PEPSYS command REDUCE/PHOT reads observational data from MIDAS table files. The user needs to construct such files from data in whatever form is available. CONVERT/PHOT helps the user build the necessary files, if raw data in either Danish or ESO standard formats are available.

**Subject:** Photometry, table files

**Syntax:** `CONVERT/PHOT`

**See also:** REDUCE/PHOT

**Note:** CONVERT/PHOT is completely interactive; no arguments are needed.

**Examples:** `CONVERT/PHOT`
This starts up an interactive session to convert a data file.

## MAKE/HORFORM *contrib/pepsys* 30-DEC-1992 ATY

**Purpose:** To simplify the task of building the horizon-obstruction table file ***hor.tbl, MAKE/HORFORM produces an appropriate blank form in the file FORM, which shows the columns needed for the actual telescope mounting, and tries to print it.

**Subject:** Photometry, forms, table files

**Syntax:** `MAKE/HORFORM`

**See also:** none

**Note:** Printing may fail for some systems, so the FORM file is left in your working directory.

**Examples:** `MAKE/HORFORM`
This prompts you for the telescope and type of mounting, and produces an empty form, to be converted to the ***hor.tbl table file.

## MAKE/PHOTOMETER *contrib/pepsys* 30-DEC-1992 ATY

**Purpose:** a) to generate the instrumental *.tbl file for a photometer; or b) to check an instrumental file that already exists.

**Subject:** photometry, table file

**Syntax:** `MAKE/PHOTOMETER`

**See also:** MAKE/STARFILE

**Note:** MAKE/PHOTOMETER operates interactively, and uses no arguments.
If you have no instrumental *.tbl file ready, it will make one for you.
If such a file already exists, it will simply display the contents in a convenient, human-readable form.

**Examples:** `MAKE/PHOTOMETER`

This starts up an interactive session to make an instrument table.

## MAKE/PLAN *contrib/pepsys* 30-DEC-1992 ATY

**Purpose:** to help an observer plan photometric observations

**Subject:** photometry, observations, planning

**Syntax:** `MAKE/PLAN`

**See also:** MAKE/STARFILE, MAKE/PHOTOMETER

**Note:** MAKE/PLAN operates interactively, and uses no arguments.
If you have telescope, instrument and star files, it will use them; otherwise, it will ask for necessary information. Non-numeric answers (other than file names) can be truncated to the first letter.
After you have answered the questions, it will produce a printable ascii file named PROUT, containing the proposed observing schedule.
Some on-line help is built in, so reply HELP or ? if a question is not clear. If you lack essential information, you can enter QUIT at any time to quit.

**Examples:** `MAKE/PLAN`

This starts an interactive session to make an observing plan.

## MAKE/STARFILE *contrib/pepsys* 30-DEC-1992 ATY

**Purpose:** The PEPSYS commands read stellar positions and magnitudes from MIDAS *.tbl files. The user needs to construct such files from data in whatever form is available. MAKE/STARFILE helps the user build the necessary files, ultimately creating the star table.

**Subject:** Photometry, table files

**Syntax:** `MAKE/STARFILE`

**See also:** MAKE/PHOTOMETER

**Note:** MAKE/STARFILE is completely interactive; no arguments are needed. It may be invoked repeatedly at different stages of the data- gathering process; if it finds you need to get more information, or prepare another file, it will exit gracefully. When you have the needed material ready, just invoke MAKE/STARFILE again and tell it what you have. It will then continue the process, until the star file is completed.

**Examples:** `MAKE/STARFILE`

This starts an interactive session to make a new table of stars.

# REDUCE/PHOT

| REDUCE/PHOT | *contrib/pepsys* | 30-DEC-1992 ATY |
|---|---|---|

**Purpose:** The PEPSYS command REDUCE/PHOT reads observational data from MIDAS *.tbl files. The user needs to construct such files from data in whatever form is available. CONVERT/PHOT helps the user build the necessary files, if raw data in either Danish or ESO standard formats are available.

REDUCE/PHOT determines and corrects for the transformations from inside to outside the atmosphere, and from instrumental to standard system. (The former is often called "extinction correction.")

**Subject:** Photometry, extinction, transformation

**Syntax:** `REDUCE/PHOT`

**See also:** CONVERT/PHOT

**Note:** REDUCE/PHOT is completely interactive; no arguments are needed.

**Examples:** `REDUCE/PHOT`
This starts an interactive session with the reduction program.

# Appendix U

# Contributed commands - romafot

# Contributed Commands

# Context: romafot

# ADAPT/ROMAFOT

## ADAPT/ROMAFOT *contrib/romafot* 08-DEC-1988 RHW

**Purpose:** Use the fit on the template image frame to derive trial values to fit a new frame

**Syntax:** `ADAPT/ROMAFOT int_tab [thres] [fac_int] [fac_sky] [fac_hol]`

`[x_size,y_size]`

| | |
|---|---|
| `int_tab` | the intermediate table with transformed coordinates by CTRANS/ROMAFOT; no default. |
| `thresh` | photometric threshold in intensity. Objects with max. intensity under this threshold are flagged and will not be considered in the fit. Nevertheless they are kept in the intermediate table (with flag set to 0) to keep the correspondence with previous registration tables. Default value is 0.0. |
| `fac_int` | factor to multiply the intensities of the objects in the intermediate table; default 1.0. |
| `fac_sky` | factor to multiply the intensity of the sky bakground in the intermediate tablep; default 1.0. |
| `fac_hol` | factor to multiply the radius of holes in the intermediate table; default 1.0. |
| `x_size,y_size` | dimensions of the new frame to fit; default same size(s) as before. |

**Note:** None

**Examples:** `ADAPT/ROMAFOT GCLINT 200 3.2 0.5 1.4 320,520`

Flag the objects with central intensity lower than 200 units in the intermediate table GCLINT; for the remainders multiply the fitted central intensity by 3.2, the sky background by 0.5 and, finally, the holes radii by 1.4. Flag the objects which, after the transformation of coordinates, exceed 320 in x and 520 in y.

ADDSTAR/ROMAFOT *contrib/romafot* 26-AUG-1989 RHW

**Purpose:** Create an artificial image identical to the original but with some subframes added at random positions

**Syntax:** `ADDSTAR/ROMAFOT in_frame out_frame [reg_tab] [cat_tab]`

`[x_dim,y_dim] [n_sub]`

| | |
|---|---|
| `in_frame` | original frame |
| `out_frame` | output frame. This is created by selecting subframes from the registration table and adding these as new components to some to the original frame at random positions. |
| `reg_tab` | registration table. This table gives position and calculated background of the objects to select the windows to be added to the original image. Default name is the input frame name extended with 'REG'. |
| `cat_tab` | output catalogue table to store the output random positions. Default name is the output frame extended with 'CAT'. |
| `x_size,y_size` | size of the subframe in x and y; default is 21,21 |
| `n_sub` | number of added subframes for each input from the registration table; default is 1. |

**Note:** A useful feature is that, if the output cat_tab created by ADDSTAR/ROMAFOT is used as input for ANALYSE/ROMAFOT. By doing so, the artificial objects can be displayed and the fraction of objects lost can eventually be estimated.

**Examples:** `ADDSTAR/ROMAFOT GCLUST GCLART WIND ? 21,21`
Select some windows from WIND.TBL, add several times these windows to the image GCLUST and create the new image GCLART. Store the random output positions in the table GCLUSTCAT.

# ADSTAR/ROMAFOT

## ADSTAR/ROMAFOT *contrib/romafot* 26-AUG-1989 RHW

**Purpose:** Create an artificial image identical to the original but with some subframes added at random positions.

**Syntax:** `ADSTAR/ROMAFOT in_frame out_frame [reg_tab] [cat_tab] [x_dim,y_dim] [n_sub]`

| | |
|---|---|
| `in_frame` | original frame |
| `out_frame` | output frame. This is created by selecting subframes from the registration table and adding these as new components to some to the original frame at random positions. |
| `reg_tab` | registration table. This table gives position and calculated background of the objects to select the windows to be added to the original image. Default name is the input frame name extended with 'REG'. |
| `cat_tab` | output catalogue table to store the output random positions. Default name is the output frame extended with 'CAT'. |
| `x_size,y_size` | size of the subframe in x and y; default is 21,21 |
| `n_sub` | number of added subframes for each input from the registration table; default is 1. |

**Note:** A useful feature is that, if the output cat_tab created by ADSTAR/ROMAFOT is used as input for ANALYSE/ROMAFOT. By doing so, the artificial objects can be displayed and the fraction of objects lost can eventually be estimated.

**Examples:** `ADSTAR/ROMAFOT GCLUST GCLART WIND ? 21,21`
Select some windows from WIND.tbl, add several times these windows to the image GCLUST and create the new image GCLART. Store the random output positions in the table GCLUSTCAT.

## ANALYSE/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** In INPUT MODE select all stars within selected subfields;
In OUTPUT MODE, look at the results of the fit operation and to select the results for final registration

**Syntax:** `ANALYSE/ROMAFOT frame [cat_tab] [int_tab] [sigma,sat]`

| | |
|---|---|
| `frame` | input frame |
| `cat_tab` | catalogue table containing the information about the objects selected manually by SEARCH/ROMAFOT or automatically by FIND/ROMAFOT. An existing catalogue table is needed in INPUT MODE. Default is the name of the frame extended with 'CAT'. |
| `int_tab` | intermediate table which will contain trial values for position, location and size of holes, intensity within the subfield of all marked stars, the subfield's size and position and background level. After the fit, trial values will be substituted by the computed parameters. An existing intermediate table is needed in OUTPUT MODE. Default is the name of the frame extended with 'INT'. |
| `sigma,sat` | Enter the sigma of the PSF and the saturation level above which the pixels are ignored. Default values are displayed. |

**Note:** The command is the central part of the ROMAFOT package to reduce stellar photometry data and has two functions. Which input tables are needed depends on which of the modes the user selects.

In INPUT MODE the command needs at least two files: the IMAGE FRAME and the CATALOGUE TABLE (containing the selected stars and created by SEARCH/ROMAFOT or FIND/ROMAFOT).
In this mode the initial coordinates are read from the catalogue table. The subfield around the star is found in the frame and is displayed. The user then marks all stars within the subfield. The intensity and position of all marked stars within the subfield together with the subfield size and background level is written to the intermediate table. Position and size of holes to ignore given regions in the subframe are also stored in the intermediate table. The intermediate table can serve also as input table, although this use is limited.

In OUTPUT MODE the command also needs two files: the IMAGE FRAME and the intermediate table (to read the computed parameters, and created by the command SELECT/ROMAFOT or by SORT/ROMAFOT).
In this mode for all the fitted subfield the program reads the image frame for the original subfield, the intermediate table for the calculated fits and displays the original subfield with the calculated fit. Together with the image, several parameters are displayed on the terminal: in particular, starting from the fourth row, five figures are reported in each row. These figures are the intensity, the local coordinates, the sigma and the fitting flag, respectively, for each object in the window. In the first row the star name is reported. In the second row one finds the record number of the intermediate table, the coordinates of the window in the image frame, the dimensions of the window in X and Y, and a flag for the fit. Finally, in the third row one finds the three parameters of the plane fitted to the sky background.

# ANALYSE/ROMAFOT

**Note:** The commands has many function in it each of which are activated by pressing a key stroke (each function has one key stroke). During execution some of these functions will ask the user for input interactively. Below a list is given with the valid key stroke and the corresponding function of the commands and the interactive input required. For more extensive description of the interactive treatment of the data the user is refered to The MIDAS User' Guide, Volume 2, Chapter 5.

A __switch between colour map and contour map display
B __enter scale factor for intensity projections (data on disk not affected)
C __enter the component to delete
D __enter the INTERMEDIATE TABLE record to display (OUTPUT mode)
E __make a hole of given radius at cursor position
F __FINISH
G __enter the hole to restore
H __display this help information
I __add a component at cursor position (trial-height is the local maximum)
J __add a component at cursor position (trial-height must be entered)
K __coordinates and pixel-value at cursor position appear on the screen
L __level above which the background at vertical cursor position appears on screen
M __enter the CATALOGUE TABLE to display an object
N __enter the INTERMEDIATE TABLE to display (INPUT mode)
P __enter identification of a component; all other components will be disabled for display or registration
Q __restore P
R __enter DELTA_X,DELTA_Y to reduce the subframe
S __enter DELTA_X,DELTA_Y to move the subframe
T __enter identification of a component: it will be disabled for display and registration. Restore with T(-n)
U __enter factor for the zero of colour scale (current 0)
V __close old files and open new ones
W __enter slope of the colour scale (larger than 0.0)
X __display isophotes
Y __enter factor (less than 1.) to get isophotes at arbitrary threshold
Z __enter "0" to display components already considered; "1" to be informed
4 __OUTPUT mode. Objects to examine: (A)ll, (M)anual or (S)elective, enter A, M or S to select one of the three modes. If (S)elective examine only special subarrays: No conv., height above ..., more than ... iterations
5 __INPUT mode. Objects to examine: (A)ll, (M)anual or (S)elective, enter A, M or S to select one of the three modes. If (S)elective examine only special subarrays: not grouped by WINDOW; height above ...
6 __enable smoothing of the displayed subframe. Data on disk will not be affected
7 __enable integral of the fitted subframe to be displayed
bar_repeat last display (with new parameters)
/ __enter new default window size
- __enable/disable hole reporting
? __enable/disable multiple registration of components
@ __enter 'R' to replace one component with the next input (using 'I' or 'J'); enter 'A' to append

For more information the user is referred to The MIDAS User's Guide, Volume 2, Chapter 5.

**Examples:** `ANALYSE/ROMAFOT GCLUST ? GCLINT`

Execute the command on the frame GCLUST using the intermediate table GCLUSTCAT. If the catalogue table does not exist the command assumes that the user wants to look at the fitted results (OUTPUT mode).

`ANALYSE/ROMAFOT GCLUSTER GCLCAT GCLINT`
Execute the command on the frame GCLUSTER using the catalogue table GCLCAT and the intermediate table GCLINT. If the catalogue table GCLCAT (e.g. created by FIND/ROMAFOT) exists the user can (INPUT mode) have a look at the windows stored in the catalogue table. After finishing the intermediate table will contain all program stars with the trial values, with dimensions and shapes. The data can be processed by the command FIT/ROMAFOT.

---

## CBASE/ROMAFOT *contrib/romafot* 12-SEP-1989 RHW

**Purpose:** Create two tables for coordinate tranformation from frame_1 to frame_2

**Syntax:** `CBASE/ROMAFOT frame_1 frame_2 [tab_1] [tab_2]`

| | |
|---|---|
| `frame_1` | first input frame |
| `frame_2` | second input frame |
| `out_tab1` | first output table for coordinate transformation; default name is TRACOO1 |
| `out_tab2` | second output table for coordinate transformation; default name is TRACOO2 |

**Note:** This command creates two tables, which are expected by CTRANS/ROMAFOT to perform the transformation of coordinates stored in an intermediate table.
On workstations the command first deletes the currently existing windows and creates two new ones in which the two frames are loaded. After the loading has completed the cursor comes and the user can select the objects first in the first and next in the second frame.
On the DEANZA the two input frames are loaded in two different channels using the split screen mode. Also here the user should first complete the object selection in the first frame and then do the selection in the second one.

**Examples:** `CBASE/ROMAFOT GCLUST1 GCLUST2`
Load GCLUST1 and GCLUST2 onto the image display; select the objects in common using the cursor.

# CHECK/ROMAFOT

## CHECK/ROMAFOT *contrib/romafot* 26-AUG-1989 RHW

**Purpose:** Examine the number of artificial stars recovered and check their photometric accuracy

**Syntax:** `CHECK/ROMAFOT cat_tab reg_tab err_mag`

| | |
|---|---|
| `cat_tab` | catalogue table containing positions and instrumental magnitudes of artificial objects. This file is created by ADDSTAR/ROMAFOT; no default name. |
| `reg_tab` | registration table. It contains photometry on the artificial image frame made using the reduced intermediate file created by FCLEAN/ROMAFOT. No default. |
| `err_mag` | limit error in instrumental magnitude. As an artificial object may fall on a pre-existing object, the simple correspondence of positions does not guarantee unambiguous identification (positions must differ less than 2 pixel to define a coincidence). No default. |

**Note:** none

**Examples:** `CHECK/ROMAFOT GCLUCAT GCLTAB 0.3`
Examine catalogue table GCLUCAT and check how many objects present here are found in the registration table GCLTAB. Two objects are coincident when their positions differ less than 2 pixel and their instrumental magnitude differ less than 0.3

## CTRANS/ROMAFOT *contrib/romafot* 14-SEP-1988 RHW

**Purpose:** Find transformation of coordinates and apply to a intermediate table

**Syntax:** `CTRANS/ROMAFOT int_tab [tab_1] [tab_2] [pol_deg]`

| | |
|---|---|
| `int_tab` | intermediate table containing coordinates which must be transformed from the system of tab_1 to that of tab_2. No default. |
| `tab_1` | table containing coordinates in the system to be transformed and created by the command CBASE/ROMAFOT. The default name is TRACOO1. |
| `tab_2` | table with coordinates in the system to transform to and created by the command CBASE/ROMAFOT. The default name is TRACOO1. |
| `pol_deg` | degree of polynomial to perform the transformation. The rms of the two fits X=f(x,y) and Y=f(x,y) is presented and the transformation is considered satisfactory if the two rms are of the order of a few tenths of pixel. Such transformation is usually achieved with degree equal 0. |
| `trans` | interactive inquiry.<br>Answer 'Y' if the rms of the transformation is small and, therefore, satisfactory. The command will then apply the transformation to the intermediate file. Default is 'N'. |

**Note:** Once the transformation is applied, the coordinates previously stored the intermediate table are overwritten. The user is therefore advised to save the intermediate table first by making copy.

**Examples:** `CTRANS/ROMAFOT GCLINT TAB1 TAB2 3`
Transform coordinates in the table GCLINT from the system in TAB1 to the system in TAB2 using a polynomial of degree 3.

DIAPHRAGM/ROMAFOT *contrib/romafot* 21-DEC-1989 RHW

**Purpose:** Make aperture photometry with fixed aperture

**Syntax:** `DIAPHRAGM/ROMAFOT frame [regi_tab] [rego_tab] ap_rad`

| | |
|---|---|
| `frame` | input image frame. |
| `regi_tab` | input registration table. The table can be created by the ROMAFOT command REGIST/ROMAFOT; default name is the frame name extended with 'REG'. |
| `rego_tab` | output registration table to receive the final results; default name is the name of the input frame extended with 'MAG'. This table is fully compatible with that created by REGISTER/ROMAFOT and contains, in particular the magnitude parameter defined as -2.5 * LOG(SUM), where SUM is the summation over all pixel values (sky subtracted). |
| `ap_rad` | radius (in pixel units) of the aperture over which the integration is performed; no default. |

**Note:** none

**Examples:** `DIAPHRAGM/ROMAFOT GCLUST GCLTAB DIA 5`
Perform aperture photometry on image frame GCLUST. Read positions of the objects in table GCLTAB and store the results in table DIA. The radius of circular area for integration is 5 pixels.

# EXAMINE/ROMAFOT

## EXAMINE/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Examine the quality of the fitted objects in the intermediate table and flag badly fitted ones

**Syntax:** `EXAMINE/ROMAFOT int_tab [hmin,hmax]`

| | |
|---|---|
| `int_tab` | intermediate table. The table is created by the command FIT/ROMAFOT; no default. |
| `hmin,hmax` | interval in intensity for the objects to examine. Default is 'all'. |
| `min,max,step` | Interactive inquiry.<br>Enter <RETURN> if you are satisfied with the plotted histogram and the fit.<br>Enter new minimum,maximum,step if you want to have it redrawn with different binning. Default values is continue. |

**Note:** The command produces two histogram distibutions from the reduced chi**2 and the semi-interquartile interval (SIQ). These data are computed for each object by the command FIT/ROMAFOT and stored in the intermediate table.
Both histograms are fitted by a gaussian function from which the sigma and the mode are computed. The user can enter the range over which these fit are made.
Hereafter a plot of chi**2 versus SIQ is produced. In the plot, on both axes, the mode in indicated with a arrow mark; the scales of the axes are in units of sigma.

By cursor control the user can indicate which objects in the plot should be considered as badly fitted and flagged. The following input given to the prompt are possible:
X: flag all stars at the right of the vertical cursor;
Y: flag all stars above the horizontal cursor;
A: flag all stars to the right of the vertical OR above the horizontal cursor;
R: restore flags;
E: EXIT;
H: display the menu.

As an example the user can issue the command 'A' to flag the objects to the right or above the cursor. Thereafter a new plot of chi**2 versus SIQ is created, excluding the flagged objects. These flagged objects can then be examined with ANALYSE/ROMAFOT to correct the input or simply ignored in the registration.

**Examples:** `EXAMINE/ROMAFOT GCLUSINT`
Examine the quality of the fitted objects in the intermediate table GCLUSINT and flag badly fitted ones in that file.

## FCLEAN/ROMAFOT *contrib/romafot* 09-AUG-1989 RHW

**Purpose:** Selects windows in the intermediate table containing objects present in the catalogue.

**Syntax:** `FCLEAN/ROMAFOT cat_tab inti_tab [into_tab]`

`cat_tab` catalogue table containing positions of objects randomly strewed on the original frame; no default name.

`inti_tab` input intermediate table containing input parameters of all the objects in a (artificial) frame (see ADDSTAR/ROMAFOT). This table is created by SEARCH/ROMAFOT and GROUP/ROMAFOT. No default.

`into_tab` output intermediate table formed by those input windows which include objects found in the intermediate table. Default is no output intermediate table.

**Note:** The command compares the objects in the catalogue with those in the windows in input intermediate table. In case the user gives an output intermediate table this tables will be filled with windows in the input intermediate containing object s present in the catalogue table. If no output intermediate table is given a selection flag is set on those windows in the input intermediate table.

**Examples:** `FCLEAN/ROMAFOT GCLCAT INTER GCLOUT`
Examine catalogue table GCLCAT and check which windows include an object present in the table INTER. Transfer these windows to the intermediate table GCLOUT.

## FIND/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Select objects using the image display

**Syntax:** `FIND/ROMAFOT frame [cat_tab]`

`frame` input image frame.

`cat_tab` name of the catalogue table which will contain the selected objects; default is the name of the frame extended with 'CAT'. This table will be used as input table for the command ANALYSE/ROMAFOT.

`name,mag,col1,col2` Interactive inquiry.
Identification of the select object, its magnitude and two colours;
default nnn,0.0,0.0,0..

**Note:** The selection of the objects from the displayed frame is done with the cursor. For that purpose switch the cursor control box and cursor 1 on and select each object by positioning the cursor and by pushing 'ENTER' on the cursor control box. Thereafter one is asked for the object name, its magnitude and two colours. These parameters are necessary in case of objects to be used as photometric standards. This command, in fact, stores three *magnitudes* in the catalogue.

After the selection procedure the catalogue table will contain the coordinates of a set of subframes. The subframes are centered on the position of the objects selected by the display cursor. The catalogue table will also contain try values for the sky background and the central intensity of selected object.

**Examples:** `FIND/ROMAFOT GCLUS GCLUSC`
Display the frame GCLUS. Select then the objects and stores the data into the catalogue table GCLUSC.

# FIT/ROMAFOT

## FIT/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Determine characteristics of stellar images by non-linear fitting.

**Syntax:** `FIT/ROMAFOT frame [int_tab] [thres,sky] [sig,sat,tol,iter]`
`[meth[,beta]] [fit_opt] [mean_opt]`

`frame` input image frame

`int_tbl` name of the intermediate table which contains the selected objects; default the frame name extended with 'INT'.

`thres,sky` photometric threshold and sky background. Data below the threshold will be ignored. Default values are those used in the commands SKY/ROMAFOT and indicated FIND/ROMAFOT.

`sig,sat,tol,iter` trial value for sigma (parameter for the width of the Gaussian or Moffet curve), the saturation, the shift tolerance, and the maximum number of iterations for each window. The defaults for sigma and saturation are the values used by previous commands (SKY/ROMAFOT); the default shift tolerance is 2 pixels, and for the number of iterations 50.

`meth,beta` non-linear fitting method to be used. One can choose between two fitting functions:
GI: a Gaussian function with inverse prop. weighting;
GU: a Gaussian function with uniform weighting;
MI,number: Moffat function with inverse proportional weighting and with beta factor number;
MU,number: Moffat function with uniform weighting and with beta factor number.
The default is MU,4.0.

`fit_opt` four character logical string consisting of 'Y' and 'N', indicating which parameters should kept fixed. The meaning of the string is as follows:
position 1: fix sigma;
position 2: fix sky background;
position 3: fix position;
position 4: allow tilted plane.
The default is 'YNNN'.

`mean_opt` character indicating if mean values of previous windows should be used as trial values for the next. One can choose between:
S: mean values for sigma should be used;
B: mean values for sky background should be used;
A: mean values for sigma and sky background should be used;
N: No mean values should be used as trials.
Default is 'N'.

**Note:** The command read the data from the intermediate table and tries to make to fit. The fit parameters will be stored back in the intermediate table on the same position. In case the fit fails or if the parameters are out of range the program will pass the message 'NO CONVERGENCE'.

The program determines the characteristics (position, width, and height) of each of the stellar images through a non-linear fitting procedure. It assumes that the PSF can be fit adequately by a gaussian or a moffat function and that a plane (optionally inclined) is a good approximation of the sky background.

The fit_opt may be used as follows: for the first parameter (sigma) give 'N' when looking for the PSF parameters and 'Y' when fitting program stars. The second parameter (plane) can in general be chosen not fixed. Only in the case of small windows which are completely filled with stellar images one can leave the sky fixed. The third parameter (position) can in most cases vary. Since in most case the tilt of the sky background over the window is negligible one can kept the background flat in principle ('N')

For more details see the MIDAS User Guide Volume 2, Chapter 5.

**Examples:** `FIT/ROMAFOT GCLUST GCLUST`
Fit the windows in the intermediate table, GCLUST, using a maximum of 50 iteration (default). Start from the beginning in the table (default), and use a Moffat function with beta=4 (default).

# GROUP/ROMAFOT



**Purpose:** Automatic grouping of objects

**Syntax:** `GROUP/ROMAFOT frame [area] [cat_tab] [int_tab] [thres]`
`[wnd_max] [end_rad,sta_rad] [wnd_perc]`

| | |
|---|---|
| `frame` | input frame. |
| `area` | area to perform the actual search. Input as can be done using the standard MIDAS notation. |
| `cat_tab` | catalogue table which contains the object parameters and created by the commmand SEARCH/ROMAFOT; default is the name of the frame extended with 'CAT'. |
| `int_tab` | intermediate table that will contain the selected objects. If the table already exists one can give the switch '/A' to append the table, or '/O' to overwrite it. If no switch has been given the user is asked for; here the default is OVERWRITE. |
| `thres` | photometric threshold below which objects are not considered as primary objects. They are however included in the windows where their influence is not negligible. Default is the threshold used by SEARCH/ROMAFOT. |
| `wnd_max` | maximum number of object per window. The maximum number of object that can be selected is 36. However a second constraint is the relation between the number of objects (MO) and holes (MH):<br>6.5*MO + 3*MH = 238.<br>MO=25 should be considered as a maximum; MO=15 is more common. The default value is 10. |
| `end_rad,sta_rad` | limiting percentages of action radius; default is 90.,100. |
| `wnd_perc` | percentage of the window to be computed. Default is 100. See the note below. |

**Note:** The option wnd_perc may be useful if the user wants wider windows (e.g. if one gets too many holes and too few pixels to sample the background), a value greater than 100 can, in fact, be given. However subframes larger than 55 by 55 pixels are not allowed.

For a more extended explanation of the command see the MIDAS Users Guide Volume 2, Chapter 5

**Examples:** `GROUP/ROMAFOT GCLUS ? CATTBL ? ? ? 15 80,100`
Group the objects in the catalogue table CATTBL and store the groups found in the default intermediate table GCLUSTINT. Take the same area of the frame on which SEARCH/ROMAFOT has worked and the same photometric threshold . The maximum number of objects per window is 15, and the limiting action radii are at the 80 and 100 percent level.

MFIT/ROMAFOT *contrib/romafot* 26-OCT-1989 RHW

**Purpose:** Determine characteristics of stellar images by non-linear fitting. The integral of the PSF is computed over the pixel area.

**Syntax:** `MFIT/ROMAFOT frame [int_tab] [thres,sky] [sig,sat,tol,iter]`

`[meth,[beta]] [fit_opt] [mean_opt] [mod_file]`

`frame` input image frame

`int_tbl` name of the intermediate table which contains the selected objects; default is the frame name with the extension 'INT'.

`thres,sky` photometric threshold and sky background. Data below the threshold will be ignored. Default values are those used in the commands SKY/ROMAFOT and indicated in FIND/ROMAFOT.

`sig,sat,tol,iter` value for sigma (parameter for the width of the Gaussian or Moffet curve), the saturation, the shift tolerance, and the maximum number of iterations for each each window. The defaults for sigma and saturation are the values used by previous commands (SKY/ROMAFOT); the default shift tolerance is 2 pixels, and for the number of iterations 50.

`meth,beta` non-linear fitting method to be used. One can choose between two fitting functions:
GI: a Gaussian function with inverse prop. weighting;
GU: a Gaussian function with uniform weighting;
MI,number: Moffat function with inverse proportional weighting and with beta factor number;
MU,number: Moffat function with uniform weighting and with beta factor number.
The default is MU,4.0.

`fit_opt` four character logical string consisting of 'Y' and 'N', indicating which parameters should kept fixed. The meaning of the string is as follows:
position 1: fix sigma;
position 2: fix sky background;
position 3: fix position;
position 4: allow tilted plane.
The default is 'YNNN'.

`mean_opt` character indicating if mean values of previous windows should be used as trial values for the next. One can choose between:
S: mean values for sigma should be used;
B: mean values for sky background should be used;
A: mean values for sigma and sky background should be used;
N: No mean values should be used as trials.
Default is 'N'.

`mod_file` Data file containing the number of intervals over each pixel and according to the table created by MODEL/ROMAFOT. The file is assumed to have extension '.dat'. Default is that PSF function is estimated over a fixed 3x3 array over each pixel.

**Note:** The difference between FIT/ROMAFOT and MFIT/ROMAFOT is that the former compares a given pixel to the value of the PSF at the centre of the pixel, the latter computes the integral of the PSF over the pixel area. The approximation used by FIT/ROMAFOT is good when the pixel size is small compared with the scale lenght of the point images.

All the comments made for FIT/ROMAFOT hold for IFIT/ROMAFOT

# MODEL/ROMAFOT

**Examples:** `MFIT/ROMAFOT GCLUST1 GCLINTER`
Fit the windows in frame GCLUST1, stored in the intermediate table GCLINTER, using a maximum of 50 iteration (default). Start from the beginning in the table (default), and use a Moffat function with beta=4 (default).

---

MODEL/ROMAFOT *contrib/romafot* 25-SEP-1989 RHW

---

**Purpose:** Compute (sub)pixel values for a model observation

**Syntax:** `MODEL/ROMAFOT [mod_file]`

`out_file` output file containing the results of the model calculation. The file extension is ".dat"; default file name is 'MODEL'.

**Note:** The command prepares a file in which the (sub)pixel values are computed according the various observing parmeters (aperature, exposure time, seeing, accuracy, magnitude, distance of the pixel from the center of the object, ...). The input paremeters are taken from the keyword INPUTR in which they should be stored as follows (using WRITE/KEY):
INPUTR(1) = seeing in arcsec
INPUTR(2) = beta
INPUTR(3) = diameter of the telescope in meters
INPUTR(4) = bandwidth in Angstrom
INPUTR(5) = total efficiency
INPUTR(6) = photons-to-ADU conversion factor
INPUTR(7) = exposure time on seconds
INPUTR(8) = pixel size in arcsec
INPUTR(9) = readout noise of the detector
INPUTR(10)= faint magnitude limit
INPUTR(11)= bright magnitude limit
INPUTR(12)= magnitude increment
INPUTR(13)= fraction of intrinsic error

**Examples:**
```
WRITE/KEY INPUTR/R/1/13 1,3,2.2,800,.3,10,1000,.35,35,23,18,.5,.1
MODEL/ROMAFOT MODEL2
```

## REGISTER/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Computes the absolute quantities and store the results in the final table

**Syntax:** `REGISTER/ROMAFOT int_tab reg_tab [wnd_opt] [obj_opt]`

| | |
|---|---|
| `int_tab` | intermediate table with the results of the fit; no default. |
| `reg_tab` | output table to receive the final results; no default. |
| `wnd_opt` | character to indicate what to store. A 'F' will register the windows for which a convergence was found; an 'A' will register all components. Default is 'A'. The components for which a 'NO CONVERGENCE' was found are registered with the instrumental magnitude set to 0. |
| `obj_opt` | character to indicate if components rejected by the command EXAMINE/ROMAFOT have to registered. 'Y' will allow the normal registration; 'N' will allow the registration but with the internal magnitude set to 0. Default is 'N'. |

**Note:** The background associated to each component in the registration table, is calculated as the sum of the lower sky background (computed by FIT/ROMAFOT) and the contribute of all the objects present in the window.

**Examples:** `REGISTER/ROMAFOT GCLUSTER REGIST F Y`
Store the results of the components for which a convergence was found in the table REGIST. Include those objects which were rejected in the command EXAMINE/ROMAFOT.

## RESIDUAL/ROMAFOT *contrib/romafot* 08-AUG-1989 RHW

**Purpose:** Compute reconstructed image and the difference with the original image

**Syntax:** `RESIDUAL/ROMAFOT in_frame out_frame diff_frame [reg_tab]`

| | |
|---|---|
| `in_frame` | original input image frame |
| `out_frame` | output image frame with the composition of all the analytical fits. |
| `diff_frame` | output image frame containing the difference between the original frame and the rec_frame. |
| `reg_tab` | registration table created by REGISTER/ROMAFOT storing the results of photometry. Default is the name of the in_frame extended with 'REG'. |

**Note:** The difference frame could be useful to detect the residuals of photometry and to add some windows with previously undetected objects to the intermediate file. Given the interactivity allowed by ROMAFOT, this command is rather abundant.

**Examples:** `RESIDUAL/ROMAFOT GCLUST RECIMA DIFFIMA ?`
Read analytical parameters from the tabel GCLUSTREG, compute their photometric effects on the original frame and create the frame RECIMA; thereafter compute the difference with the original frame and put the result in the frame DIFFIMA.

# SEARCH/ROMAFOT

## SEARCH/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Do the actual search for objects above a certain threshold

**Syntax:** `SEARCH/ROMAFOT frame [sky_tab] [cat_tab] [area] [psf_par] [thres] [height]`

| | |
|---|---|
| `frame` | input frame. |
| `sky_tab` | table containing sky values (from SKY/ROMAFOT); default is the frame name extended with 'SKY'. |
| `cat_tab` | output table which will contain the object found; default is the frame name extended with 'CAT'. |
| `area` | area to perform the actual search. Input as can be done using the standard MIDAS notation. |
| `psf_par` | sigma, beta of the PSF and saturation. If the PSF is described by a gaussian, beta should be set at 0. Default values are 3.,4.,15000. The default can be used if the user is not interested in automatic grouping. |
| `thres` | option and value for the photometric threshold (above the sky). Two inputs may be given:<br>C,val: the threshold will be constant with value val;<br>R,val: the treshold will be set at val times the sky standard deviation. |
| `height` | height at which the star is considered not to influence the results photometrically. The action radius associated to each object, AR, and, consequently, the size of holes is strongly dependent on this figure. An interactive check with ANALYSE/ROMAFOT is recommended after GROUP/ROMAFOT and before FIT/ROMAFOT. |

**Note:** The command produces a list of (sometimes) thousands of objects. Inspection of these using the command ANALYSE/ROMAFOT might be instructive. To proceed it is recommended to use the command GROUP/ROMAFOT. Interactive inspection of the catalogue table is possible using the command ANALYSE/ROMAFOT.

**Examples:** `SEARCH/ROMAFOT GCLUST`
Find objects above a constant threshold over the whole area of the image.

`SEARCH/ROMAFOT GCLUST ? ? ? ? ? R,3`
Find objects above a relative threshold of 3 standard deviations of the sky background (S is asked interactively); these sky values are supposed to be stored in the table GCLUSTSKY. The object found will be put into the table GCLUSTCAT.

## SELECT/ROMAFOT *contrib/romafot* 07-AUG-1989 RHW

**Purpose:** Select objects and store the positions in the intermediate table

**Syntax:** `SELECT/ROMAFOT frame [int_tab] [wnd_size]`

| | |
|---|---|
| `frame` | input image frame |
| `int_tab` | name of intermediate table that will contain the selected objects; default is the frame name with extension 'INT'. |
| `wnd_size` | size of the area over objects will be fitted (see FIT/ROMAFOT); default is 21 pixels both in the x and y direction (21,21) |
| `name,mag,col1,col2` | interactive inquiry. identification of the select object, the magnitude, and two colours; default nnn,0.0,0.0,0.0 |

**Note:** The selection of the objects from the displayed frame is done with the cursor. For that purpose switch the cursor control box and cursor 1 on and select each object by positioning the cursor and by pushing 'ENTER' on the cursor control box. Thereafter one is asked for the object name and its colors. These parameters are necessary in case of objects to be used as photometric standards, this command, in fact, stores three *magnitudes* in the intermediate table. If this is not the case, the default values can be used.

After the selection procedure the intermediate table will contain a set of subarrays with dimensions according to the specified fit window. The subframes are centered on the position of the objects selected by the display cursor. Also, the table will contain try values for the sky background and the central can be passed directly to FIT/ROMAFOT.

**Examples:** `SELECT/ROMAFOT GCLUST ? 30,30`
Select the object from frame GCLUST and stores the data into the intermediate table GCLUSTINT with array size of 30 by 30 pixels.

`SELECT/ROMAFOT GCLUST INTTBL 20,20`
Select the object from frame GCLUST and stores the data into the intermediate table INTTBL with array size of 20 by 20 pixels.

# SKY/ROMAFOT

| SKY/ROMAFOT | *contrib/romafot* | 07-AUG-1989 RHW |
|---|---|---|

**Purpose:** Determines intensity histogram and sky background in selected areas

**Syntax:** `SKY/ROMAFOT frame [sky_tab] [area] [nrx,nry] [min,max]`

| | |
|---|---|
| `frame` | input image frame |
| `sky_tab` | name of the output table which will contain the values for the sky background; default is the frame name extended with the three characters 'SKY'. |
| `area` | area over which the background should be calculated. Input can given using the standard MIDAS notation. Default is the whole frame. |
| `nrx,nry` | number of subdivisions in X and Y to create the grid of rectangular cells where the trial value of the sky background is determined. Default is 5,8 |
| `min,max` | minimum and maximum between which a histogram of the frame pixel values is calculated (see Note) Default values are the cut values in the descriptor (LHCUTS(1) and LHCUTS(2)). If these values are identical the minimum and maximum in the frame are taken. |

**Note:** First, a histogram of the pixel values between min and max is created, and the maximum in the histogram distribution is determined. A second window is then taken with a width of 10of the min,max range and centered around the pixel value that corresponds with the peak in the histogram distribution. The background is determined taking all values into account that lie within this window.
The maximum number of cells is limited to 256.

**Examples:** `SKY/ROMAFOT GCLUSTER ? [100,300:400,500] ? 500,2000`
Determine the sky background in 5 by 8 cells in the area enclosed by the pixels corners (100,300) and (400,500). The output table will be named GCLUSTERSKY. Only values between 500 and 2000 should be taken into account in determining the background level.

# Appendix V

# Contributed commands - surfphot

# Contributed Commands

# Context: surfphot

# COMPUTE/FCOEFF

## COMPUTE/FCOEFF *contrib/surfphot* 08-Sep-1985 PJG

**Purpose:** Compute fourier coefficients of azimuthal profiles of spiral galaxies

**Syntax:** `COMPUTE/FCOEFF infram orient rin,rout,rstep outtab`

| | |
|---|---|
| `infram` | input frame |
| `orient` | center of galaxy in x, center of galaxy in y, position angle, inclination. The center of the galaxy should be given in pixel units. |
| `rin,rout,rstep` | inner radius, outer radius, step in radius. Input is assumed to be in pixel units. |
| `outtab` | output table. For each of the radii the resulting coefficients will be stored in this table. |

**Note:** The program first extracts the azimuthal profiles, given the orientation parameters. The profiles are then fourier transformed and the coefficient are stored in the table. The number of coefficients is 7. i.e. the 6th harmonic is included.

**Examples:** `COMPUTE/FCOEFF myframe 100,100;32,20 20,60,10 mytab`

## COMPUTE/GRID *contrib/surfphot* 20-Aug-1985 DB

**Purpose:** Create image with distorted but otherwise rectangular and evenly spaced grid plus a table with the points defining the grid.
The grid image is displayed and the grid table also plotted.

**Syntax:** `COMPUTE/GRID angle`

| | |
|---|---|
| `angle` | distortion angle of the grid (in degrees), defaulted to 1.E-5, which gives a slight exaggeration of the distortion typically found in PCD and ST-FOC image |

**Note:** Size of both image (510 x 510 pixels) and table (81 rows) are fixed. For most tests it may be advisable to EXTRACT a subframe of the image.
In order to simulate point spread functions with a larger FWHM, the image may be FILTERed.

**Examples:** `COMPUTE/GRID 0.00002`
Create image GRID.bdf with distorted grid and the table GRID.tbl with the points defining the grid.
Load the image GRID.bdf and plot the table GRID.tbl .

COMPUTE/SKY *contrib/surfphot* 05-JUL-1989 AL/RHW

**Purpose:** Compute the sky background and restitute the frame

**Syntax:** `COMPUTE/SKY inframl infram2 caltab method sky_factor`

| | |
|---|---|
| `inframl` | first input frame |
| `infram2` | second input frame; if method = 3 then infram2 = inframl |
| `caltab` | calibration table containing calibration coefficients for thr characteristic curve. This table can be created with CONVERT/DTOM |
| `method` | method to be used:<br>1 = PDS scan after 14/4/1984 (descriptor O_TIME set to 1984.5<br>2 = PDS scan before 14/4/1984<br>3 = only one input frame, second frame is equal to first frame; only desciptor SKYBGR is update; this is the default;<br>0 = same as 1 but no pixel replacement |
| `sky_factor` | sky_factor; default 1.4 |

**Note:** With the two input frames A and B (where B is a subset of A) and the calibration table, containing the coefficients the following will be done:
a. A flat sky background will be through the intensity level lower than the highest intensity level in the histogram of frame A multiplied by sky_factor. The background is determined inside 8 rectangualar areas surrounding frame B and within two concentric circles;
b. get the PDS density population of the upper 5 and 10 percent in the intensity distribution in B;
c. Determine the centroid of these pixel distributions.
If inframl = infram2 only part of the operations is done.

**Examples:**
```
COMPUTE/SKY GAL1 GALSUB GALTAB 1 2.0
```

# FILTER/FILL

## FILTER/FILL *contrib/surfphot* 10-OCT-1988 DB

**Purpose:** (Partially) fill up low-flux (below threshold) pixels with flux from high pixels in neighborhood box of radius rx,ry.

**Syntax:** `FILTER/FILL inframe outframe rx,ry threshold`

| | |
|---|---|
| `inframe` | name of input image |
| `outframe` | name of result frame |
| `rx,ry` | x and y radius of neighborhood box; default is 1,1 |
| `threshold` | flux threshold, below and up to which pixels are filled up; default is 0 |

**Note:** 1.) One possible application of this command is the preparation of images for subsequent deconvolutions (DECONVOLVE/IMA or REBIN/DECONVOLVE) that require a low, but non-negative background. (The final, however not flux conserving, step will probably have to be REPLACE/IMAGE.) Generally, it is an efficient means to reduce background noise.
2.) The algorithm searches for low pixels (i.e. with a flux below the threshold). Then it searches the box around this central pixel for high pixels. In a third step, it searches the neighborhood box around each of these high pixels for other (secondary) low pixels. Finally, the command redistributes flux from the high pixels to the central pixel. All neighboring high pixels contribute the same amount which is determined from the following rules:
a) The central low pixel does not receive more flux than necessary to reach the threshold.
b) A given high pixel contributes the same amount to all low pixels in its respective neighborhood box.
c) No high pixel will end up with a flux below the threshold.
3.) The total flux is strictly conserved. The mean level of the background will increase the more strongly the more crowded the field is.
4.) The command can be iterated by using the result of the previous iteration as input for the next one. In the ith iteration, it will be the flux left over in the ith-ranking pixel (counted in ascending order) which limits the amount of flux that is re-shuffled. For larger box dimensions, the difference between subsequent iterations is, therefore, much smaller.

**Examples:** `FILTER/FILL noisy filtered`
Frame noisy.bdf is filtered using a box of dimension [2*1+1,2*1+1] by as far as possible filling up negative-flux pixels from immediately neighboring high pixels (if any). Results will be in frame filtered.bdf.

FIND/PAIR *contrib/surfphot* 20-Oct-1991 RHW

**Purpose:** Match (pair) two point sets with coordinates in table files INA and INB. The common set of points form a new table file OUTA with added columns, excluded points are appended to the common set,

**Syntax:** `FIND/PAIR intab1 intab2 outtab columns [errors] [coo_sys]`

| | |
|---|---|
| `intab1` | first input table; no default. |
| `intab2e` | second input table; no default. |
| `outtab` | output table, containing common data point of the two input tables; no default. |
| `columns` | columns number of the first input table containing the x and y coordinates, followed by the columns in the second input table; no default. |
| `errors` | estimated separations in X and Y between 1st and 2nd tables. The first two values contain the maximum residual distance after coordinate translation (in X and Y), the second two vaules the maximum residual distance after coordinate rotation. Default is 0.0,0.0,2.0,2.0 |
| `coo_sys` | coordinate transformation option:<br>0: coordinates in their own systems (default);<br>1: 2nd system transformed to 1st system. |

**Note:** Both object sets should have almost the same scale and orientation.
It is possible to apply PAIR several times, building up merged tables of increasing complexity, containing three or more original subtables. Note, that at any time only two sets of X,Y coordinates are fully treated; X,Y coordinates associated with other subtables may gradually be lost (set to NULL element). Therefore, it is advisable to always keep one fixed (+extended) set of X,Y coordinates (fixed column nos) as the first system (1st table for example) in the next pairing.
Second set of labels have characters 14-16 replaced with column numbers in order to avoid repeated label names; embedded blanks are replaced with underscore (_).
In determining the pair of objects, the common translation vector is estimated as the mode of histogram of X,Y separations. In the second iteration a small coordinate rotation is made.
Not matched points are added at the end of the output table, left or right coming from the 1st or 2nd table, respectively.
Maximum number of objects in each table is limited to 50000.

**See also:** Tables commands

**Examples:** `FIND/PAIR B346 U346 BU 10,11,10,11 0.,0.,40.,30.`
Combines table files B3460120 and U3460005 into table BU using X,Y column nos 10,11 and 10,11 for 1st and 2nd input file, assumed X,Y translation 0.,0. between 1st and 2nd system, tolerances 40. and 30. units (here microns), where the 1st tolerance assumes no rotation nbetween the systems. In a second iteration a small (max 1 degree) corrective rotation and a very minor rescaling is applied and the matching is repeated with a new more restrictive tolerance.
The following message may be displayed (mean translation within brackets):
B346 - U346: ( 56.193 12.663) 260 + 211 objects => 180 pairs
residual: DX= 0.12231E+02 + 0.697E-05 * X + -0.133E-02 * Y
rotation: DY= -0.27403E+02 + 0.135E-02 * X + 0.132E-03 * Y
B346 - U346: ( 53.832 8.699) 260 + 211 objects => 177 pairs

# FIND/POSINC

## FIND/POSINC *contrib/surfphot* 08-Sep-1988 PJG

**Purpose:** Find the position angle and inclination of a galaxy

**Syntax:** `FIND/POSINC inframe x_pos,y_pos rin,rout,rstep`

| | |
|---|---|
| `inframe` | input frame |
| `x_pos,y_pos` | center of galaxy in x, center of galaxy in y. |
| `rin,rout,rstep` | inner radius, outer radius, step in radius. |

**Note:** The program estimtes the position angle and inclination of a galaxy using the variation of the 2nd fourier harmonic. The coordinate input is assumed to be in pixel units. The results are written in the keyword OUTPUTR(1:4). These results are: x centre position, y centre position, the position angle and the inclination (the last both in degrees).

**Examples:** `FIND/POSINC myframe 100,100 20,60,10`

## FIT/BACKGROUND *contrib/surfphot* 08-MAY-1990 RHW

**Purpose:** Compute coefficient of a 2-dim polynomial fit of the sky background. Optional create frames with the background and/or the background subtracted input frame.

**Syntax:**

```
FIT/BACKGROUND outframe = inframe(s) [deg,it] [clp1,clpn] [skew]
[outbck]

FIT/BACKGROUND outframe = inframe(s) [coef] [outbck]

FIT/BACKGROUND inframe(s) [deg,it] [clp1,clpn] [skew] [outbck]

FIT/BACKGROUND inframe(s) [coef] [outbck]
```

| | |
|---|---|
| `outfram` | output frame, i.e. the background subtracted input frame; no default allowed. |
| `inframe(s)` | input frame and, if desired, the input mask and to be specified as `inframe,inmask`. No default for the input frame and no masking wanted. |
| `outbck` | name of the output background frame. By default no background frame is generated. |
| `deg,it` | degree of the polynomial to fit and the maximum number of iterations. Default values are 3,10. |
| `clp1,clpn` | clipping parameter for thr first and clipping parameter for the following iterations. Default is 1.0,2.5. |
| `up_skew` | upper limit for the acceptable skewness. Default is 0.05. |

**Note:** The coefficients are calculated by an iterative least-squares fitting of the pixels in a rebinned input frame. If a predefined 'masking frame' is given, only not masked pixels are taken into account.

**Examples:**

```
FIT/BACKGROUND ccdbck = ccd
```

Compute the background in the frame ccd and store the background corrected frame in ccdback. Use all default values.

```
FIT/BACKGROUND ccd,ccdmsk 2,20 2,4 ?  ccdbck
```

Compute the background in the frame ccd and store the background corrected frame in ccdback. Use the frame ccdmsk to get the pixels which should not be taken into account for the background determination. The fit should be of the order of 2 with 20 iterations at maximum. Clip values should be 2 for the first and 4 for the following iterations. Use the default skewness value.

# FIT/ELL1

| FIT/ELL1 | *contrib/surfphot* | 14-Mar-1991 RHW |
|---|---|---|

**Purpose:** An image is searched for intensity points within a certain intensity range and within a radius to the center.

**Syntax:** `FIT/ELL1 inframe outframe l_iso,h_iso x_cen,y_cen max_rad`

| | |
|---|---|
| `inframe` | input frame |
| `outframe` | output frame with isophotal points and fitted ellipse |
| `l_iso,h_iso` | low and high isophotal levels |
| `x_cen,y_cen` | x and y centre in world coordinates |
| `max_rad` | radius around center to be surveyed; default is value used in the previous run |

**Note:** The ellipse centre (x and y), the semi-major diameter, the semi-minor diameter and the position angle (degrees, o on positive x axis increasing to positive y axis) will be saved in the descriptor ELLPAR attached to the output frame.
The maximum survey radius should be slightly larger than the semi-major diameter of the fitted ellipse - a too large survey diameter of the fitted ellipse includes to many spurious pixels at large distances, producing a too large (and round) fitted ellipse.

**See also:** LOAD/IMAGE

**Examples:**
```
FIT/ELL1 INFRAME FITFRAM 1,5 98.5,101.1 20
```

FIT/ELL2 *contrib/surfphot* 17-Feb-1992 RHW

**Purpose:** An image is searched for intensity points within a certain intensity range and within a radius to the center.

**Syntax:** `FIT/ELL2 inframe pol_opt iso_tol iso_levels [center[ [radius]`

`[sky_level]`

| | |
|---|---|
| `inframe` | input frame |
| `polar_opt` | 0, if polar frames should be made (first run) 1, polar frames already exist (repeated runs) |
| `iso_tol` | isophotal tolerance in promille of iso_levels |
| `iso_levels` | isophotal levels you want to fit (max = 10) the format can be low:high:incr or lev1,lev2,.. or a combination of the two separated by a comma |
| `center` | x,y-coordinate of center of object; default is the value used in the previous run |
| `radius` | radius around center to be surveyed; default is value used in the previous run |
| `sky_level` | value to be added to the iso_levels before a fit is made to the data points; default is the value in the previous run |

**Note:** The input image should have been cleaned from disturbing objects (stars) in the region of interest. The center of the object can be calculated using the command CENTER/GAUSS.
The ellipses are fitted with respect to the predefined center. At high isophotal levels the fit will be affected by the seeing and it will become round, the position angle being undefined. At low isophotal levels (< 5 - 2 of a round object. With these natural restrictions the code allows to trace isophotal twisting.
Points that were found and lay exactly on the ellips can not be discriminated from the ellips.
The parameters of the ellips are communicated to the user.

The command creates two polar coordinate frames containing the polar coordinates radii and angles of every pixel of the input frame. The polar coordinates of the points found are used to fit an ellips with a predefined center. Every rerun of the program (with polar_opt=0) will delete the preexisting scratch files of the polar coordinates.

The command also produces an image called ELLIPS with the points used for the fit and with the actual ellips determined. This image can be used for display. Different isophotal levels will displayed in different colours. In the examples below the use of this file together with the input data file is demonstrated.

**Examples:**
```
FIT/ELL2 INFRAME 0 10 10:50:10 -356.,-456 1000 125
```
and run again bad solution for the 50_isophote changing the tolerance

```
FIT/ELL2 INFRAME 1 30 50
```
or run again changing the radius of the search area

```
FIT/ELL2 INFRAME 1 10 50 300
```
etc. In this way one can interactively steer the fit towards optimal solutions.

```
FIT/ELL2 INFRAME 0 10 10:50:10,70,90 -356.,-456.  1000 0
LOAD/IMAGE INFRAME 0
LOAD/IMAGE ELLIPS OVERLAY
SET/OVERLAY
```

# FIT/ELL3

```
FIT/ELL2 INFRAME 1 10 10:50:10,70,90 -356.,-456.  1000 0
LOAD/IMAGE INFRAME 0
LOAD/IMAGE ELLIPS OVERLAY
BLINK 0 OVERLAY
```

## FIT/ELL3 *contrib/surfphot* 10-Dec-1990 MS+RHW

**Purpose:** fit ellipses to the isophotes of an object in a 2-dim. frame

**Syntax:** `FIT/ELL3 inframe outframe [step] [x,y] [low,high] [min,max] [opt]`

| | |
|---|---|
| `inframe` | input frame |
| `outframe` | output frame containing the model reconstructed from the fit |
| `step` | spacing of the isophot levels. The levels C(n) are logarithmically equidistant: C(n+1)=C(n).10**(-step). Default is 0.01. |
| `x,y` | estimated x and y centre of the innermost ellipse. If not given, the command will read elements 5 and 6 of the real descriptor CENTER. |
| `low,high` | lowest and highest isophote level. Default display cuts. |
| `min,max` | min, max position angle (degrees, from x-axis counter clockwise) of the sector to be excluded from the fit. Default is 0,0. |
| `opt` | option to enable (Y) or not enable cleaning (N). See below; default is 'N'. |

**Note:** The last two parameters min,max and opt are aids in case that the isophotes are disturbed: a single major disturbation can be excluded by excluding the sector which contains it; multiple minor perturbations are removed by comparing the values of pixels at opposite position angles and selecting the minimum (opt = Y).
An estimate of the x,y center of the innermost isophote may be obtained with the CENTER/GAUSS command which optionally also produces the CENTER descriptor.
In addition a table 'outframe'.TBL is produced which contains the fitted parameters: the isophote levels in column :Z; the major and minor axes in columns :A and :B, respectively; the position angle of the major axis in :PA; the center of the ellipses in :X and :Y; and the noise in :NOISE.
The algorithm is based on the formulas of R. Bender and C. Moellenhof published in Astron. and Astrophys. 177,71 (1987).

**See also:** COMPUTE/TABLE, LOAD/IMAGE

**Examples:** `FIT/ELL3 m87 model 0.05 ?  ?  20.,30.  Y`
This will fit ellipses to the isophotes of an object located at CENTER coordinates in frame M87 and write result into frame MODEL. The isophote levels are logarithmically spaced with 0.05. The lowest and highest levels are defined by display cuts. An angle section between 20 and 30 degree will be excluded from fit. The cleaning option is enabled.

## FIT/POSINC *contrib/surfphot* 09-Feb-1988 PJG

**Purpose:** Fit the position angle and inclination to 2nd and 4th harmonic

**Syntax:** `FIT/POSINC infram orient rin,rout,rstep region`

| | |
|---|---|
| `infram` | input frame |
| `orient` | center of galaxy in x, center of galaxy in y, position angle, inclination. The center of the galaxy should be given in pixel units. |
| `rin,rout,rstep` | inner radius, outer radius, step in radius. Input is assumed to be in pixel units. |
| `region` | enter for the position angle and inclination where the fit has to be done |

**Note:** The fit is done within the region given such that the phases of the 2nd and 4th harmonic have the minimum sigma.

**Examples:** `FIT/POSINC myframe 100,100,32,20 20,60,10 10,20`

## INTEGRATE/ELLIPS *contrib/surfphot* 29-Apr-1991 RHW

**Purpose:** Integrate pixel intensities within ellipse in 2-D image

**Syntax:** `INTEGRATE/ELLIPS frame [ellips_param] [flag]`

| | |
|---|---|
| `frame` | input image frame |
| `ellips_par` | ellipse parameters: center X, center Y, semi-major diameter, semi-minor diameter, position angle (degrees, 0 on x-axis, incr towards the y-axis) Default are the values used in the previous run. |
| `flag` | 0 or 1. With a 1 the ellipse parameters are obtained from the descriptor ELLPAR of the input frame; default is 0. |

**Note:** The program does a simultaneous search for points within a slightly larger ellipse, determines the differential increase in pixel area, ellipse area and flux. A linear interpolation is applied to get flux from pixels covering same total area (approx same region) as the nominal ellipse.
The result is stored in the descriptor ELLPAR(1) to ELLPAR(6). The flux (integral over all pixel values is stored in the descriptor ELLPAR(6)

**Examples:** `INTEGRATE/ELLIPS myframe 250,312,50,30,45`

# NORMALIZE/IMAGE

## NORMALIZE/IMAGE *contrib/surfphot* 17-OCT-1983 KB

**Purpose:** normalize and truncate a frame

**Syntax:** `NORMALIZE/IMAGE in out trunc_vals control_vals`

| | |
|---|---|
| `infram` | input frame |
| `outfram` | output frame |
| `trunc_vals` | low + high cutoff for normalized image (i.e. the background is approx. = 1.0) to take care of the noise and the sensitivity of the detector, defaulted to: no truncation |
| `control_vals` | see Inventory program... |

**Note:** none

**Examples:** `NORMALIZE/IMAGE CCD0012 CCDN12 0.77,4.8`
Normalize the frame CCD0012 and write the result into the frame CCDN12. Use 0.77 and 4.8 for the low and high cut values.

## REBIN/DECONVOLVE *contrib/surfphot* 04-OCT-1988 LL,DB

**Purpose:** Rebin an image linearly in space and simultaneously deconvolve it with user-supplied point spread function. (Reference: L.B. Lucy: 1974, Astron. J. 79, 745-754)

**Syntax:** `REBIN/DECONVOLVE frame [psf] result zoom_x,zoom_y [n_iter]`

| | |
|---|---|
| `frame` | name of input image |
| `psf` | name of point spread function image (default: psf) |
| `result` | name of result frame |
| `zoom_x,zoom_y` | zoom factors in X and Y |
| `no_iter` | number of iterations, defaulted to 1 |

**Note:**

1.) It is highly recommendable first to remove the background to the lowest possible level, however NOT thereby introducing pixels with negative fluxes. See also commands FIT/FLAT_SKY, SUBTRACT/SKY, and FILTER/FILL.

2.) The PSF must have odd numbers of pixels in both X and Y (which however do not have to be same for both coordinates); it must be centered on the central pixel. The step sizes must be the ones desired for the result frame. If the empirically determined PSF (not zoomed) is n pixels wide, it is recommended to construct from it the required zoomed (by a factor z) PSF to have (n+2)*z pixels.

3.) The zoom factors must be odd integers and consistent with the ratios of the step sizes of in-frame and psf. Zoom factors less than unity are illegal. For zoom_x = zoom_y = 1, the command is functionally equivalent to DECONVOLVE/IMAGE (but slower!).

4.) The result frame must have the same step size as the point spread function. The number of pixels must be compatible with the number of pixels of the input frame and the zoom factors.

5.) If MID_WORK contains a file 'result.bdf', this file will be interpreted as an initial approximation of the result. Using the command n times with n_iter=1 therefore has the same effect as using it once with n_iter=n except that intermediate results can be inspected.

6.) The command can be used particularly profitably in order to improve the appearance of images with sparsely sampled PSF's where the smearing effects of conventional rebinning unwanted. It may also help to bring out faint features more strongly against the background. Depending on various factors, a larger number of iterations does not necessarily imply better results.

7.) The contents of each pixel is interpreted as flux density (e.g. surface brightness), i.e. flux per unit area in world coordinates. This means that the sum over all output pixels equals the sum over all input pixels times zoom_x times zoom_y.

8.) Edge effects are suppressed by extrapolating the frame beyond its edges at a constant level and with the flux of the last pixel observed.

9.) Large frames in combination with large zoom factors may exceed the available virtual memory.

**Examples:** `REBIN/DECO observed seeing true 3,5 2`

Expand image observed.bdf by factors 3 and 5 in X and Y, respectively. At the same time deconvolve with psf seeing.bdf using 2 iterations. The result will be stored in frame true.bdf which will be newly created if not already existing. Issuing the very same command again will execute iterations Nos. 3 and 4 on frame true. The net effect is the same as the one of REBIN/DECO observed seeing true 3,5 4

# RECTIFY/IMAGE

## RECTIFY/IMAGE *contrib/surfphot* 20-AUG-1985 DB/KB

**Purpose:** Geometrically rectify a distorted direct image (e.g. obtained with an image tube).

**Syntax:** `RECTIFY/IMAGE in out table [nrep] [deconvol_flag]`

| | |
|---|---|
| `in` | input frame |
| `out` | output frame |
| `table` | table with the following columns<br>#1 : theoretical "x"-coordinates of reseau marks<br>#2 : theoretical "y"-coordinates of reseau marks<br>#3 : measured "x"-coordinates of reseau marks<br>#4 : measured "y"-coordinates of reseau marks<br>All coordinates are expected in units of pixels of the image. |
| `nrep` | additional sub-stepping factor (cf. "Note" below);<br>default: 1, maximum: 5 |
| `deconvol_flag` | deconvolution flag. Y=YES, N=NO. Default: N |

**See also:** RECTIFY/SPECTRUM, COMPUTE/GRID

**Note:** A very CPU-time intensive program!
The photometric accuracy per pixel is thought to be better than about 1The flux conservation is perfect if also nearest neighbours are considered. At the expense of further CPU time consumption, the said 1This seems justified only for observations of the highest S/N. The programme routinely performs a simple "deconvolution" assuming a fixed point spread function (PSF) of about one pixel which appears appropriate for a wide range of applications. This can be suppressed (deconvol_flag = N) for test purposes or low-resolution data, i.e. data with a FWHM of the PSF of a few pixels, but even for high-resolution data the results are not bad. With deconvol_flag = N (and nrep = 1), the execution time is reduced by a factor of 5 !

**Examples:** `For a demo use the command COMPUTE/GRID (no parameters).`

This will create a distorted grid image (GRID.bdf) as well as the relevant table (GRID.tbl) containing the theoretical and measured reseau marks.
You can then try the algorithm on these test inputs, e.g.:
RECTIFY/IMAGE GRID outgrid GRID Load the result image outgrid.bdf to see the effect of RECTIFY/IMAGE .

## SUBTRACT/SKY *contrib/surfphot* 01-MAR-1991 DB

**Purpose:** Compute most probable sky from histogram of sky reference area within image. Subtract this sky from input frame.

**Syntax:** `SUBTRACT/SKY inframe outframe nx,ny flag`

| | |
|---|---|
| `inframe` | name of input image |
| `outframe` | name of result frame |
| `nx,ny` | dimensions (in pixels) of neighbourhood box (cf. below), default: 1,1 |
| `flag` | Y or N to indicate whether or not the central pixel be included in the computation of the flux in the box. Default: N |

**Note:** 1.) This command may be useful in situations where, as for deconvolutions, a low but non-negative background is required. (Note, however, that the total flux is not strictly conserved.) It can also lead to a dramatic reduction of the noise in the background.
2.) Inframe must have descriptors HISTOGRAM, STATISTIC, WINDOW_FROM, and WINDOW_TO present as set by command STATISTIC/IMAGE. The histogram must be representative of the sky in the entire image. That is, the histogram must not be the one of the full frame! Large-scale structures in the background must have been removed, e.g. using FIT/FLAT_SKY. The first and the last histogram bin are assumed to be excess bins as defined for STATISTIC/IMAGE. The bin width should be chosen such as to sample the distribution of background pixel values reasonably well. The quality of the results can be very sensitive to the high cut which may, therefore, have to be refined after a first trial.
3.) The command works as follows (see also Baade, D., Lucy, L.B.: 1989, Proc. 1st ESO/ST-ECF Data Analysis Workshop, p. 169): The sky reference histogram is scaled up to the frame's full area. The bin-wise ratio to the (internally calculated) frame's actual histogram then represents the probability that a pixel with a flux within a given bin is due to sky only. For bins with this probability being unity, all pixels with the corresponding signals are zeroed. For other bins, the probability predicts how many pixels owe their signal to the sky only; let this number be NZERO. Rank all pixels belonging to the given histogram bin according to the average flux in their neighbourhood of (2*nx+1)*(2*ny+1) pixels. Set the first (lowest) NZERO pixels this sequence to zero. From the rest subtract the value of all bins between (i.e., excluding) the lower excess bin and the one considered.
4.) Pixels in the high excess bin will have the expectation value of all non-excess bins subtracted. From pixels in the low excess bin, the lower cutoff value of the histogram will be subtracted. Therefore, these latter pixels will be recognizable by their negative fluxes. If desired, they can be zeroed by means of REPLACE/IMAGE.

**See also:** FILTER/FILL, FIT/FLAT_SKY

**Examples:** `SUBTRACT/SKY back manipulated 2,2 N`

On the basis of descriptors HISTOGRAM, STATISTIC , WINDOW_FROM, and WINDOW_TO generated by command:
STATISTIC/IMAGE back (plus suitable parameters!)
model the sky in frame back.bdf and subtract the model sky from the image. Result will be in frame manipulated.bdf. The ranking of pixels is done on the basis of the flux in a neighbourhood of 5x5 pixels but not including the central pixel of this box.

# Appendix W

# Contributed commands - tsa

# Contributed Commands

# Context: tsa

# AOV/TSA

**AOV/TSA** *contrib/tsa* 15-SEPT-1992 A. Schwarzenberg-Czerny

**Purpose:** Compute analysis-of-variance periodogramme. The expected value of the periodogramme for pure noise is 1, for uncorrelated observations and 'ncorr', for correlated observations, where 'ncorr' is an average number of correlated observations. The the periodogramme divided by its expected value has Fisher-Snedecor probability distribution F(order-1,nobs/ncorr-1), where 'nobs' is number of observations and 'order' is defined below. For reference see M.N.R.A.S. 241, 153.

**Syntax:** `AOV/TSA intab outima start step nsteps [order] [cover]`

| | |
|---|---|
| `intab` | name of input table, it must contain columns :TIME and :VALUE in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE. |
| `outima` | name of output image; its first row contains the AOV periodogramme, the second row contains quality flags indicating the quality of phase coverage for a given frequency. Meaning of the flags:<br>0 - all phases covered,<br>1 - underfilled phase bins occurred,<br>2 - empty bins occurred<br>Statistics of bad bins is listed onto terminal. |
| `start` | start frequency of the periodogramme, in inverse units of :TIME |
| `step` | frequency step of the periodogramme, in inverse units of :TIME |
| `nsteps` | number of frequencies of the periodogramme |
| `order` | number of phase bins to be employed, 1 < order < 101. |
| `cover` | number of overlapping bin covers to be employed. Each consecutive cover is shifted by 1/(order*cover) in phase. 0 < cover; default cover = 2. |

**See also:** SHOW/TSA, DFT/TSA, SCARGLE/TSA

**Note:** By default 'intab', 'outima', 'start', 'step', 'nsteps' and 'order' retain their values from the last use of TSA commands, unless explicitly specified. See SET/TSA for details of this feature.
For smooth light curves use low 'order', e.g. 4 or 3 for best sensitivity.
For many observations and light curves with sharp features (e.g. pulses, eclipses) use phase bins of width comparable to that of these features. This will boost sensitivity above that attainable with the Scargle method.
Note that phase coverage at and near 0 frequency is notoriously poor for any observations. Hence underpopulated or empty bins may occurre. The summary statistics is printed onto the terminal and for details see the outima quality row.

**Examples:**
```
AOV/TSA LCURVE PERIODG 0.01 0.01 100 4
```

```
AOV/TSA ? ?  0.2 0.0001
```
This second command may be used after the first example to inspect the frequency interval from 0.2 till 0.21 at higher resolution.

BAND/TSA *contrib/tsa* 15-SEPT-1992 A. Schwarzenberg-Czerny

**Purpose:** Sampling times of observations are analysed and a suitable frequency band and resolution are derived. The keywords defining the frequency grid STARTTSA, STEPTSA and NSTEPS are set. This command requires observations sorted in ascending order in time.

**Syntax:** `BAND/TSA intab [maxobs]`

| | |
|---|---|
| `intab` | name of input table, it must contain columns :TIME and :VALUE in DOUBLE PRECISION. |
| `maxobs` | maximum number of observations analysed for separation. The default value maxobs=0 causes use of all observations. This causes unnecessary waste of time for uniform or randomly distributed observations with only few large gaps. In such a case supply a number of initial observations whose spacing is representative for the whole data. |

**See also:** SHOW/TSA, POWER/TSA, AOV/TSA, SCARGLE/TSA

**Note:** none

**Examples:**

```
BAND/TSA LCURVE
```

```
BAND/TSA LCURVE 100
```

# COVAR/TSA

| COVAR/TSA | *contrib/tsa* | 15-SEPT-1992 | A.Schwarzenberg-Czerny |
|---|---|---|---|

**Purpose:** Compute discrete covariance function for unevenly sampled data. The binned covariance function is returned with its errors. For reference see Ap.J. 333, 646.

**Syntax:** `COVAR/TSA intab1 intab2 outtab start step nsteps [scale] [funct]`

| | |
|---|---|
| `intab1` | name of the table containing 1st set of observations with their variances. The table must contain columns :TIME :VALUE and :VAR in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE. |
| `intab2` | same for 2nd time series. |
| `outtab` | name of the output table containing the covariance function. |
| `start` | start value of time lag in the output table, in units of time or log time depending on value set for the parameter scale. |
| `step` | step of time lags in the table, in units of time or log time depending on value set for the parameter scale. |
| `nsteps` | number of time lags in the table |
| `scale` | type of lag scale to be used (LINear or LOGarithmic). The default value is LIN. |
| `funct` | type of output function: C - correlation function (default), C(lag)= <x(t)*y(t+lag)>, S - structure function, S(lag)= <(x(t)-y(t+lag))**2>/2, where C(lag)=sqrt(Var[x]*Var[y])-S(lag), Var[] is signal variance and is assumed that both signals have null mean. |

**See also:** SHOW/TSA, DELAY/TSA

**Note:** This command can be used for the calculation of the autocovariance function.

**Examples:**

```
COVAR/TSA DATA1 DATA2 COV 1.  0.1 31
```

```
COVAR/TSA DATA1 DATA1 AUTOCOV
```

returns the autocovariance function if time lag grid was defined by a previous call

DELAY/TSA *contrib/tsa* 15-SEPT-1992 A. Schwarzenberg-Czerny

**Purpose:** Compute chi squared-time lag function for two time series. Minima of the chi2 indicate a possible physical association of two time series delayed by the corresponding lag. Individual measurements must be given with their variances. This command requires as input the autocovariance function of the observations in analytical form. The algorithm is described in Ap.J. 385, 404. We use here the full algorithm (fitting both mean and difference at the same time), not the abrigated version fitting difference only. However, the results of the two versions do not differ significantly according to our and author's tests. Note that the whole method (not the code) is not yet finally tested, thus use it with care. Both algorithms involve inverting the observation covariance matrix of size equal to the number of observations, for each time lag, a quite time consuming operation and thus are slow.

**Syntax:** `DELAY/TSA intab1 intab2 outtab start step nsteps func parm`

`intab1` name of the table containing 1st set of observations with their variances. The table must contain columns :TIME :VALUE and :VAR in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE.

`intab2` same for 2nd time series.

`outtab` name of the output table containing chi squared vs. time lag.

`start` start value of time lag in the output table

`step` step of time lags in the table

`nsteps` number of time lags in the table

`func` code for the analytical form of the autocorrelation function of the observations. The user may choose to use one of the predefined formulae (LIN,... IPO) suppling just parameters or to specify USi, where i=0,1,...9 and supply his/her own code for the function TSADELUR0. The options are

```
LIN ACF(x) = A + B*x (default)
POL ACF(x) = A + B*x +C*x**2 +....  (up to 12 coeff)
POW ACF(x) = A + B*x**C
EXP ACF(x) = A + B*exp(C*x)
LOG ACF(x) = A + B*LOG(C*x)
IPO ACF(x) = A + B/(x-x0) + C/(x-x0)**2 + ...
where A=PARM(1), B=PARM(2), ...., x0=PARM(12),
except for IPO where x0=PARM(1), A=PARM(2), ...
URi ACF(x) = user defined function TSADELURi
```

The head of the user supplied function must be the following:

```
DOUBLE PRECISION FUNCTION TSADELURi(TLAG,PARM)
DOUBLE PRECISION TLAG,PARM(12)
.........................
```

The computed value of TSADELURi must depend only on ABS(TLAG) and (optionaly) on PARM(1), PARM(2),.....

`mode` code for the algorithm to be used: NOR - Both mean value and difference between the sets are determined by covariance matrix weighted least squares. This new improved algorithm is recommended (default).

`parm` parameters for the function used 1) from command line: a,b,c,d,....(max.12) 2) from keyword: "K(iname)", e.g. KINPUTR Default values are 0,1,0,0,...

**See also:** SHOW/TSA, COVAR/TSA, INTERPOLATE/TSA

# DELAY/TSA

**Note:** The analytical form of ACF function can be created using output of COVAR/TSA and MIDAS FIT package.
Since this command can take long time to execute, call it first for just one time lag to get an estimate of the computation time.
A progress report is printed every one 10th of the total number of lags. It lists lag, std.dev. of the fit (its square constitutes chi**2 per degree of freedom), fitted mean and difference between the samples, and their errors.

**Examples:** `DELAY/TSA DATA1 DATA2 CHI2 0.  1 1 EXP 0,1,-0.25`
gives you an estimate of the computation time just for 1 lag

`DELAY/TSA DATA1 DATA2 CHI2 ?  ?  200 UR7 KINPUTR`
computes the function for 200 lags. The parameters are the same than before (0,1,-0.25) and the command uses the user defined function

```
DOUBLE PRECISION FUNCTION TSADELUR7(TLAG,PARM)
DOUBLE PRECISION TLAG,PARM(12)
TSADELUR7=PARM(1)+PARM(2)*EXP(PARM(3)*ABS(TLAG))
END
```

The results should be the same as before.

INTERPOLATE/TSA *contrib/tsa* 15-SEPT-1992 A. Schwarzenberg-Czerny

**Purpose:** Interpolate an unevenly sampled series using its analytic covariance function. The algorithm is described in Ap.J. 385, 404. Individual measurements must be given with their variances. This command requires as input the autocovariance function of observations in analytical form. Note that this method (not the code) is not yet finally tested, thus use it with care.

**Syntax:** `INTERPOLATE/TSA intab outtab func parm`

`intab` name of the table containing observations with their variances. The table must contain columns :TIME :VALUE :VAR in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE.

`oname` same for the output time series. The corresponding table must exist and contain columns :TIME :VALUE :VAR. Only the values in :TIME are used for input.

`func` code for analytical form of the autocorrelation function of the observations. The user may choose to use one of the predefined formulae (LIN,... IPO) suppling just parameters or to specify USR and supply his/her own code for the function TSAINTUSR. The options are

```
LIN ACF(x) = A + B*x (default)
POL ACF(x) = A + B*x + Cx**2 +....  (up to 12 coeff)
POW ACF(x) = A + B*x**C
EXP ACF(x) = A + B*exp(C*x)
LOG ACF(x) = A + B*LOG(C*x)
IPO ACF(x) = A + B/(x-x0) + C/(x-x0)**2 + ...
where A=PARM(1), B=PARM(2), ....,
except for IPO where x0=PARM(1), A=PARM(2),...
USR ACF(x) = user defined function TSADELUSR
```

The head of the user supplied function must be the following:

```
DOUBLE PRECISION FUNCTION TSAINTUSR(TLAG,PARM)
DOUBLE PRECISION TLAG,PARM(12)
.......................
```

The computed value of TSAINTUSR must depend only on ABS(TLAG) and (optionally) on PARM(1), PARM(2),.....

`parm` parameters for the function used 1) from command line: a,b,c,d,....(max.12) 2) from keyword: "K(iname)", e.g. KINPUTR Default values are 0,1,0,0,...

**See also:** DELAY/TSA, COVAR/TSA, SHOW/TSA

**Note:** The analytical form of the ACF function can be created using the output of COVAR/TSA and the MIDAS FIT package.

**Examples:**

```
INTERPOLATE/TSA DATA1 DATA2 EXP 0,1,-0.25
```

```
INTERPOLATE/TSA DATA1 DATA2 USR KINPUTR
```

We should get the same results as before assuming that the parameters 0,1,-0.25 are saved in keyword INPUTR defining the user function:

```
DOUBLE PRECISION FUNCTION TSAINTUSR(TLAG,PARM)
DOUBLE PRECISION TLAG,PARM(12)
TSAINTUSR=PARM(1)+PARM(2)*EXP(ABS(TLAG)*PARM(3))
END
```

# NORMALIZE/TSA

| NORMALIZE/TSA | *contrib/tsa* | 15-SEPT-1992 | A.Schwarzenberg-Czerny |
|---|---|---|---|

**Purpose:** Normalize mean and (optionally) variance of a table column to 0 and 1, respectively. Other columns are simply copied.

**Syntax:** `NORMALIZE/TSA iname oname column mode`

| | |
|---|---|
| `intab1` | name of input table |
| `outtab` | name of a similar output table |
| `column` | column affected |
| `mode` | mode of normalization,<br>M - mean only (default)<br>V - both mean and variance |

**See also:** none

**Note:** See SET/TSA for defaults of intab1 and outtab.

**Examples:**
```
NORMALIZE/TSA LCURVE ? :TIME

NORMALIZE/TSA LCURVE ? :VALUE V
```

POWER/TSA *contrib/tsa* 15-SEPT-1992 A.Schwarzenberg-Czerny

**Purpose:** Compute discrete power spectrum for uneven sampling by slow method. This command is recommended as departure point for the analysis of sampling (spectral window) and signal. However, since the power spectrum has no well defined statistical properties we recommend use of AOV/TSA and/or SCARGLE/TSA for assessing the significance of features in the periodogrammes.

**Syntax:** `POWER/TSA intab outima start step nsteps`

| | |
|---|---|
| `intab` | name of input table, it must contain columns :TIME and :VALUE in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE, except for window function calculation. |
| `outtab` | name of output image; its first row contains the power spectrum and the second row contains the window function. |
| `start` | start frequency of the periodogramme, in inverse units of :TIME |
| `step` | frequency step of the periodogramme, in inverse units of :TIME |
| `nsteps` | number of frequencies of the periodogramme, in inverse units of :TIME |

**See also:** SHOW/TSA, AOV/TSA, SCARGLE/TSA

**Note:** By default 'intab', 'outtab', 'start', 'step' and 'nsteps' retain their values from the last use of TSA commands, unless explicitly specified.
For detection of smooth signals use SCARGLE/TSA.
Sensitivity of power spectrum to signals with sharp features is poor, use AOV/TSA instead.

**Examples:**
```
POWER/TSA LCURVE PERIODG 0.01 0.01 100
```

```
POWER/TSA ? ?  0.2 0.0001
```
You may run this second command after the first example to inspect the frequency range 0.2 to 0.21 at higher resolution.

# SCARGLE/TSA

| SCARGLE/TSA | *contrib/tsa* | 15-SEPT-1992 | A.Schwarzenberg-Czerny |
|---|---|---|---|

**Purpose:** Compute Scargle periodogramme for unevenly spaced observations by a slow method. This command is recommended for the statistical evaluation of the significance of smooth (sinusoidal) oscillations. Scargle statistics for uncorrelated observations obeys an exponential probability distribution, with expected value of 1. For observations correlated in groups of size 'ncorr', divide value of Scargle statistics by 'ncorr'. For reference see Ap.J. 263, 835.

**Syntax:** `SCARGLE/TSA intab outima start step nsteps`

| | |
|---|---|
| `intab` | name of input table, it must contain columns :TIME and :VALUE in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE. |
| `outima` | name of output image; its first row contains Scargle periodogramme and second row contains quality factors. Where factors are small, Scargle periodogramme differs from power spectrum considerably. For unit factors Scargle periodogramme differs from power spectrum only by normalization to unit variance. |
| `start` | start frequency of the periodogramme, in inverse units of :TIME |
| `step` | frequency step of the periodogramme, in inverse units of :TIME |
| `nsteps` | number of frequencies of the periodogramme, in inverse units of :TIME |

**See also:** SHOW/TSA, SET/TSA, AOV/TSA, POWER/TSA

**Note:** By default 'intab', 'outima', 'start', 'step' and 'nsteps' retain their values from the last use of TSA commands, unless explicitly specified.
For the detection of signals with sharp features use AOV/TSA instead.

**Examples:**
```
SCARGLE/TSA LCURVE PERIODG 0.01 0.01 100
```

```
SCARGLE/TSA ? ?  0.2 0.0001
```
You may run this command after the first example to study the frequency range from 0.2 till 0.21 in more detail.

## SET/TSA *contrib/tsa* 15-SEPT-1992 A.Schwarzenberg-Czerny

**Purpose:** Set explicitly global keywords for Time Series Analysis context TSA. The keyword values are subsequently reset by other TSA commands to current values of the corresponding parameters. In this way the last value remains default value for the next command.

**Syntax:** `SET/TSA keywordname=value (keywordname=value...)`

`keywordname` name of a keyword

`value` value to be substituted for the keyword

The following keywords are defined and save last values of corresponding command parameters:
Key name: Content - Command parameters
INROOT1: Last input file - [intab,intab1,inima,inascii]
INROOT2: Last secondary input - [intab2]
OUTROOT: Last output file - [outtab,outima]
STARTTSA: Grid start - [start]
STEPTSA: Grid step - [step]
NSTEPS: Grid size - [nsteps]
ORDERTSA: Order - [order]
The corresponding command parameters if specified explicitly are saved in the keywords. If not specified, they use keyword value as default.

**See also:** SHOW/TSA, COMPUTE/KEY

**Note:** Keywords name should not appear in the command line and command parameters should not appear in SET/TSA or COMPUTE/KEY.

**Examples:**
```
SET/TSA ORDERTSA=2 STEPTSA=0.01
```

## SHOW/TSA *contrib/tsa* 15-SEPT-1992 A.Schwarzenberg-Czerny

**Purpose:** Show global keywords for Time Series Analysis context TSA. The keywords values are set when explicit values of parameters are specified in TSA commands or with COMPUTE/KEY.

**Syntax:** `SHOW/TSA`

`Parameters` no parameters required

**See also:** COMPUTE/KEY

**Note:** none

**Examples:**
```
SHOW/TSA
```

# SINEFIT/TSA

SINEFIT/TSA *contrib/tsa* 15-SEPT-1992 A.Schwarzenberg-Czerny

**Purpose:** Fit sine (Fourier) series, optionally by nonlinear iterations with correction of frequency f. The series has the form: S(t)=C(0)+C(1)*sin(Ph)+C(2)*cos(Ph)+...C(2*[order])*cos([order]*Ph), where phase Ph=2*pi*f*t. The fitted values of the coefficients C(i) and their errors are displayed in the order of appearance in the formula. The last fitted entry is frequency. The frequency is not adjusted in the very first iteration so carries 0 nominal error. A line with details of the fitted ephemeris follows. The phase at m.t. stays for the phase at mean time. The frequency and this phase are orthogonal (principal or uncorrelated) results of the fit and thus are recommended for use.

The command constitutes also a sort of filter: takes observations from the input column :VALUE and returns them prewhitened (i.e. detrended) with the fitted series again in output :VALUE. So it can be used to perform a CLEAN analysis by removing consecutive frequencies manually step by step. Alternatively it can be used to remove trends from data. The command works in two modes: start iterations or continue iterations, depending on parameter iter.

**Syntax:** `SINEFIT/TSA intab outima freque order iter`

`intab` name of input table containing time series in columns :TIME and :VALUE in DOUBLE PRECISION. For numerical reasons it is advisable to subtract mean from :VALUE on start. Note that the same value has to be added to the result constant (0 frequency) coefficient.

`outab` name of similar output table containing :TIME, original values in :ORIG_VALUE, fitted values in :FIT and residuals from the fit in :VALUE. Outab and intab could be the same.

`freque` base frequency, default and continuation value is STARTTSA. Since this routine is not intended for general frequency search but is intended for refining frequencies as well as all sorts of strange detrending and filtering applications, it remains user responsibility to make sure that his initial frequency guess and its final fit are plausible. Use SCARGLE/TSA, AOV/TSA or POWER/TSA with zoomed frequency scale for that purpose.

`order` order of Fourier series (0 - only constant, 1 - pure sine of base frequency, 2 - base and 1st harmonic, etc. ... Default and continuation value is ORDERTSA

`iter` number of iterations to be performed. If your input and output file names differ you can use the same command with frequency replaced by ? to continue the the iterations. For experts only: use iter < 0 if you wish to continue abs(iter) iterations stepwise in exactly the same manner as they would occurre in a single SINEFIT/TSA call. In the latter case be sure that none of keywords OUTROOT, STARTTSA and ORDERTSA were changed between consecutive iterations by other MIDAS commands. In practice no other TSA commands nor MIDAS commands changing OUTPUTD can be used between negative iterations. Default +1.

**See also:** none

**Note:** The coefficients C(i) of the fitted series as defined above are returned in OUTPUTD(i).

**Examples:**
```
SINEFIT/TSA LCURVE RESID 0.205 1
```
will remove a given frequency from the data

```
SINEFIT/TSA RESID RESID 0.333
```
will prewhiten with another frequency

```
SINEFIT/TSA LCURVE RESID 0.205 2 4
```

will fit frequency, using base and 1st harmonic Fourier series

If necessary, you can continue 3 more iterations with

```
SINEFIT/TSA ? ?  ?  ?  3
```

continues 3 more iterations. The result will differ slightly from SINEFIT/TSA LCURVE RESID 0.205 2 7

---

TABLE/TSA *contrib/tsa* 15-SEPT-1992 A.Schwarzenberg-Czerny

---

**Purpose:** Convert ASCII table into MIDAS table

**Syntax:** `TABLE/TSA inascii outtab type mxcol`

| | |
|---|---|
| `inascii` | name of input ASCII file with extension .dat. The file should contain data written in columns of FORTRAN free format consistent with the conversion type. Text lines in this file if any are displayed and otherwise ignored. |
| `outtab` | name of MIDAS output table (default - 'inascii') |
| `type` | conversion type for numerical data: R,D or I (default - D). |
| `maxcol` | maximum number of columns scanned in 'inascii', need not be the number of actually present columns (default - 10). |

**See also:** CREATE/TABLE, COMPUTE/TABLE

**Note:** The default column labels :RVALi (DVALi or IVALi), (i=1,...) can be changed with NAME/COLUMN to those requested by a particular command.

**Examples:**

```
TABLE/TSA table produces table.tbl in D(ouble) format

TABLE/TSA ascii data D 20 produces table with up to 20 D columns
```

# WIDTH/TSA

WIDTH/TSA *contrib/tsa* 15-SEPT-1992 A. Schwarzenberg-Czerny

**Purpose:** This command finds the strongest line in the frequency band of 'centre'+/-'width'/2 and determines its characteristics, by fitting its profile, among others. The profile used at the moment is a cosine bell. The same band is used to find the continuum level, so it shouldn't be too narrow. The STARTTSA keyword is set to the line centre frequency and OUTPUTD returns width and level values, as listed.

**Syntax:** `WIDTH/TSA inima [width] [centre]`

| | |
|---|---|
| `inima` | name of input frame, it must contain a periodogramme in its first row, in DOUBLE PRECISION. |
| `width` | full width of the frequency band searched for the strongest line to be measured, in units of frequency. Default - the whole periodogramme. Note that this band is used to define the continuum, so should not be too narrow. However, a too broad band causes time loss on determination of the continuum level. |
| `centre` | central frequency of the band searched for the strongest line to be measured. Default - the centre of the periodogramme. |

**See also:** SHOW/TSA, POWER/TSA, AOV/TSA, SCARGLE/TSA

**Note:** You input the f u l l width of the frequency band, while the command prints h a l f widths of the line, sort of a frequency error estimate. The error of central frequency fit tells only how well the theoretical profile fits the line and by no means constitutes an estimate of accuracy of line frequency.
Note that the formal errors of the fit are multiplied by the square root of the number of periodogramme points falling within one half width of the line. This takes care of a possible correlation of the residuals (same sign) over that width, and so provides a more meaningful error estimate than usual least squares.
Do not unnecessary run the command over intervals containing too many points, since than it becomes slow. Use plot/row to identify the frequency of the line.

**Examples:** `WIDTH/TSA PERIODG 1 23`
analyse a line centred around frequency 23

`WIDTH/TSA PERIODG`
analyse the strongest line in the whole spectrum. However, the line width estimation could be quite slow in this case.

# Appendix X

# Contributed commands - astromet

# Contributed Commands

# Context: astromet

# ASTROMETRY/COMPUTE

| ASTROMETRY/COMPUTE | *contrib/astromet* | 17-Oct-1994 | OH, RHW |
|---|---|---|---|

**Purpose:** Convert the coordinates from the measured xy ot RA,Dec and vice versa, using the transformation transformation parameters computed by ASTROMETRY/TRANSFO.

**Syntax:** `ASTROMETRY/COMPUTE mes option out trail`

| | |
|---|---|
| `mes` | table containing the XY measurements; It must have been previously processed by ASTROMETRY/TRANSFO in order to have the descriptors with the transformation parameters set. |
| `option` | specifies what should be done with the data:<br>1: Calculate (A,D) for plate center (assuming that the first measurements correspond to the plate crosses) and several (X,Y)-values from the mes table.<br>2: Calculate (A,D) for several (X,Y)-values from the mes table. This is the most used option (and the default).<br>3: Calculate (A,D) for one (X,Y)-value entered at the keyboard. In this case, mes table is used only to get the transformation parameters, and is not actually red.<br>4: Calculate (X,Y) for one (A,D)-position entered at the keyboard. In this case, mes table is used only to get the transformation parameters, and is not actually red.<br>5: Calculate (X,Y) for several (A,D)s in disc file (not implemented in Midas... the input file must be ASCII). |
| `out` | output table, where the computed coordinates are stored. This table can be used as standard stars table in case a multistep astrometric calibration is needed. |
| `trail` | Flag indicating if the measurements correspond to the end of the trails (Y), eg left by a moving object or not (N); default is N. |

**See also:** ASTROMETRY/TRANSFO, ASTROMETRY/EDIT, ASTROMETRY/POS1

**Note:** ASTROMETRY/COMPUTE computes the A,D from X,Y using directly the transformation parameters, while it obtain X,Y from the A,D in an iterative manner.

NOTE THAT THE EQUINOX OF THE COMPUTED COORDINATES IS THE SAME AS THE STANDARD STARS CATALOGUE (which is NOT necessarilly the same as its epoch). 2000.0 in case the PPM is used.

Option 5 is not fully implemented in MIDAS; it works like in the former standalone version, using an ASCII input file.

**Examples:** `ASTROMETRY/COMPUTE mesxy342 2 sec342`
Convert the measured X,Y from mesxy342 in A,D, and store the results in sec342. The objects were not trailed.

ASTROMETRY/EDIT *contrib/astromet* 17-Oct-1994 OH, RHW

**Purpose:** Delete/undelete the astrometric standards used for the transformation, and optionally displays (slowly) the map of the residuals.

**Syntax:** `ASTROMETRY/EDIT std plot`

| | |
|---|---|
| `std` | table containing the standars stars coordinates. This table MUST have been processed previously by ASTROMETRY?TRANSFO |
| `plot` | P to plot the map of the residuals, N otherwise (default). |

**See also:** ASTROMETRY/TRANSFO, ASTROMETRY/EDIT, ASTROMETRY/POS1

**Note:** After having (optionally and slowly) plotted the map of the residuals, the programs ask for the Nr of a star to be deleted/restaured. This must be the sequence Nr in the Midas table and NOT the identifier of the star! Type 0 to exit.

**Examples:** `ASTROMETRY/EDIT ppm3`
Edit the standards (without plotting the residuals). A typical dialogue would like like:
Star to be deleted/resaured? (0 to exit) 0
5 [delete star 5]
Star to be deleted/resaured? (0 to exit) 0
43 [delete star 43]
Star to be deleted/resaured? (0 to exit) 0
123 [delete star 123]
Star to be deleted/resaured? (0 to exit) 0
5 [restaure star 5]
Star to be deleted/resaured? (0 to exit) 0
0 [exit]

# ASTROMETRY/POS1

| ASTROMETRY/POS1 | *contrib/astromet* | 17-Oct-1994 | OH, RHW |
|---|---|---|---|

**Purpose:** Interactive procedure for the POS1 astrometry package

**Syntax:** `ASTROMETRY/POS1`

**See also:** ASTROMETRY/TRANSFO, ASTROMETRY/EDIT, ASTROMETRY/COMP

**Note:** This procedure corresponds to Richards West's original POS1 program. The input/output are made compatible with MIDAS, but the algorithm has not been changed (as it proved to be extremely accurate, this would have been masochism).
While the original POS1 was doing everything in one programme, the MIDAS version has been split in 3 steps:
1* read the measurements and standard stars, and compute the transformation parameters (this step is performed by the command ASTROMETRY/TRANSFORM).
2* edit the standard star table to remove/restore some stars (this step is performed by the command ASTROMETRY/EDIT)
3* actually compute the converted coordinates (this step is performed by the command ASTROMETRY/COMPUTE)

Steps 1 and 2 can be iterated until a satisfactory result is obtained. The whole process can be performed in an iterative way; the user is asked for all the parameters.

**Examples:** `ASTROMETRY/POS1`
Start an interactive astrometric reduction.

## ASTROMETRY/TRANSFO *contrib/astromet* 17-Oct-1994 OH, RHW

**Purpose:** Compute the astrometric transformation parameters of a data set.

**Syntax:** `ASTRO/TRANS std mes center pla,cat schmidt,blink tol xterm,yterm std`

`std` table containing the standars stars coordinates. It must contain the following columns:
:PPM Identifier (<=6digit intg) (must be called :PPM even if the standards are not from that catalogue);
:R_A Right Ascension in deg;
:DEC Decl. in deg;
:MAG Magnitude;
:PMA Proper Motion in RA in arcsec/year (NOT time_second);
:PMD Proper Motion in Dec in arcsec/year;

A convenient way to produce this table is to use StarCat, with the MIDAS table output option, in the PPM catalogue;

`mes` table containing the XY measurements; It must contain the following columns: :IDENT Identifier; must contain a number, leading letters (eg, PPM) are ignored :Xcen X measurement :YCEN Y Measurement

`center` coordinates of the center, either Ah,Am,As,Ddeg,D',D" = the actual coordinates of the center (the decli. between -1 and 0 are handled properly), or MEAN to take the barycenter of the measured standards (useful for CCD astrometry; don't use on plates).

`pla,cat` epoch of the plate (date at which it was taken) and reference epoch of the catalogue (in decimal year); note that cat is NOT necessarilly the equinox of the catalogue coordinates. For the PPM, pla=2000.

`schmidt,blink` instrument flags schmidt = Y if a (curved) schmidt plate was measured, N otherwise (plane detector, like a CCD) blink = Y if the plate was measured with the La Silla blink (which has a funny unit coding), N otherwise (normal measuring machine dealing correctly with negative units, eg OPTRONICS, or MIDAS CENTER/GAUSS)

`tol` tolerance on the measurements RMS (in measurement unit, eg micron or pixel); meaningfull only in case of multiple measurements like those made on the OPTRONICS. The points outside the tolerance range are delete.

`xterm,yterm` terms to be used for the astrometric calibration in X and Y. Set the digit to 1 to select the term, to 0 to ignore it. The terms are: X Y XY X**2 Y**2 X**3 Y**3 XY**2 YX**2 For a first try, or a CCD, use 111000000,111000000; for a small region of a Schmidt plate, use 111110000, and for a full Schmidt plate, 111111111. Note that standard stars in sufficient number must be measured, at least 2–4 time the number of terms.

`std` selection of the standard flag. A takes all the standards (eg, first try), U only the undeleted ones (subsequent try). Use residual.prg to select/unselect the standards.

**See also:** ASTROMETRY/EDIT, ASTROMETRY/COMPUTE, ASTROMETRY/POS1

# ASTROMETRY/TRANSFO

**Note:** This program computes the parameters using a least-square algorithm. It is a upgraded version of Richard West's standalone program. The accuracy MIDAS is excellent; it has been tested on standard stars, and in real life on many objects including comet Shoemaker-Levy 9, leading to very accurate impact times.

ASTROMETRY/TRANSFO computes only the transformation parameter; use ASTROMETRY/COMPUTE to perform the actual coordinate conversions. Use ASTROMETRY/EDIT to edit the standard stars (delete/reset) and to display a map of the redidual.

ASTROMETRY/POS1 is a small procedure asking for the paramaters of ASTROMETRY/TRANSFO, ASTROMETRY/EDIT, ASTROMETRY/EDIT in an interactive way, simulating the former stand-alone pos1.

**Examples:** `ASTROMETRY/POS1`
Start an interactive astrometric reduction.

`ASTROMETRY/TRANSFO POS1A ppm3 p300 17,21,2.2,-33,56,40 -`
1969.83,2000. yn 3 111110000,111110000 A Reduction of a Schmidt plate (not measured with the La Silla blink): first iteration (All stars take, only 5 terms). The standards are in ppm3.tbl, the xy measurements in p300.tbl. The plate was taken on 1969.83, centered on 17,21,2.2 -33,56,40. The epoch of the catalogue is 2000; the tolerance is set to 3 micron.

`ASTROMETRY/EDIT ppm3`
Edit the standards

`ASTROMETRY/TRANSFO ppm3 p300 17,21,2.2,-33,56,40 1969.83,2000. yn 3`
111111111,111111111 U Second iteration of the reduction: all the terms are set, and only the undeleted standards are taken for the transformation.

`ASTROMETRY/EDIT ppm3 P`
Edit the standards, and display a map of the residuals

`ASTROMETRY/TRANSFO ppm3 p300 17,21,2.2,-33,56,40 1969.83,2000. -`
yn 3 111111111,111111111 U Third iteration of the reduction: all the terms are set, and only the undeleted standards are taken for the transformation.

`ASTROMETRY/COMPUTE p300 2`
Computes the RA Dec of the measured objects.