TRIUMF	UNIVERSITY OF ALBERTA EDMONTON ALBERTA	
	Date 3-March-1991	*File No* TRI-DNA-91-2
Author P. W. Green		*Page* 1
Subject Summary of NOVA Commands		

NOVA Command Summary

The following contains an alphabetical list of NOVA commands. Some of them will be redundant in the final version of the system - they are part of the test package I am using to develop the system - so if you can't see a use for a particular command (or don't understand it) don't worry too much - it will probably vanish before you ever have to use it.

In these descriptions, the term *item* refers to anything that the user might define (variable, spectrum, condition, OPSEQ label, ...).

NOVA command lines are as "free-format" as possible (I hate systems where parameters must be separated by EXACTLY one space, or only commas are allowed - especially if the rules are not consistent for all commands). Parameters in NOVA commands are separated with commas and/or one or more blanks or tabs. Qualifiers (which begin with the **/** character) need not (although may be) separated with blanks or tabs.

Most (not all) commands support *wild cards* in user names. The standard VMS convention applies - the character '%' matches *exactly* one character in the name and the character '*' matches *any number* (including zero) of characters in the name.

In most cases, command lines are "executed" in strict left to right order. Thus, qualifiers should *precede* the parameters to which they apply. For example, to obtain a "full" listing of a variable, you must enter

LIST/FULL variable

rather than

LIST variable/FULL

This is a little different than you might be used to (see, for example, the discussion of **DEF1D** and **DEF2D** below).

All commands (and qualifiers) may be abbreviated to their shortest *unique* form. Thus**, DEL, DELE, ...** are all legal substitutes for the **DELETE** command.

VERY IMPORTANT!

It is *guaranteed* that in this and all subsequent versions of the system, all commands will be unique within the first 4 characters (i.e. a 4 character abbreviation will never cause confusion). Especially in command files, it is a good idea to always use at least 4 characters since future releases of NOVA might introduce new commands which conflict with command abbreviations shorter than 4 characters.

User Commands

NOVA commands are intended to be as "non-cryptic" as possible. With a few exceptions (*ZALL* is one that comes to mind) it is (I think) fairly obvious from the command word only what the command is supposed to do. This, plus the fact that abbreviations to commands must be unique, means that sometimes more typing is required than in other systems. As one particular example, many systems use the command word **D** to mean display - this is nice and short and therefore appropriate for something that you commonly use. This won't work in NOVA, because D could mean not only *DSP* but also *DUMP, DELETE, DEF1D, DEF2D, ...*

To allow the user to customize the system to his liking as much as possible, NOVA allows *User defined command words.* Any legal name (12 or fewer characters with the first one alphabetic) can be defined to be equivalent to any character string in a command. The only restrictions on the use of User defined commands are:

1. The command can only be used at the *beginning* of a command string. That is, when the system receives a command line to be executed, it looks at the *first word only* on the line to see if it is a user defined command. If it is, this word is replaced with the appropriate string, and the command line is then processed.

2. This replacement process is *not recursive* (that is, if the first word is replaced with a user command string, the process is not repeated). Thus, user commands may not be defined in terms of other user commands.

3. The total length of any command line in the system is limited to 255 characters.

Thus, for example, if you commonly use the command DSP/PAGE=4 to plot 4 spectra on one screen, you could define a user command called D4 to do this.

D4 := DSP/PAGE=4

and then say

D4 S1 S2 S3 S4

Or, if you just like to use D for DSP, you could define

D := DSP

The syntax for defining a user command is similar to that used for defining a symbol in VMS:

command := replacement_string

The operator ':=' rather than just '=' tells NOVA that this is the definition of a user command and not a variable (note, though, that ':==' does NOT work).

Note that User commands are stored in the tables along with everything else, so they are destroyed with a LOAD command. If you want to make use of user commands, it is probably best to keep them all in a command file, and then you can restore all of your definitions after a load with @*file.* You can use the LIST/USER command to see what user commands are currently defined.

General

!Comment

Any command line which begins with an exclamation point is treated as a Comment, and is ignored by NOVA.

CRTL-C

Most commands can be terminated by typing CTRL-C (*NOT* CTRL-Y - that terminates your NOVA session). For example, a lengthy LIST command can be aborted by typing CTRL-C, and you will be returned to the NOVA Command Line Interpreter.

Editing Command Lines

Any command line may be edited (using the cursor keys to position the cursor, as with VMS DCL) before pressing RETURN to execute the command. As well, any of the most recent 20 commands may be recalled, edited (if desired) and then re-executed by just pressing RETURN. Press PF1 to see the commands which may be recalled, PF2 for HELP and PF4 to invoke a simple desk calculator.[a]

The command editing facility is not available if you have SET INPUT/REMOTE.

@filename

Begin reading commands from the file *filename.* @-files may be nested up to 10 deep. An @-file is terminated (and hence further command lines will be read from the next higher level) by the physical end of file. If *filename* does not contain an extension, .DAT is assumed.

NOVA_STARTUP

When you first type NOVA, the system searches for a file named *NOVA_STARTUP.DAT* in your default directory (or you may use the logical name NOVA_STARTUP to redirect this assignment), and if it finds it, executes it as an @-file. You could use this, for example, to set your operating environment (e.g. you might want LONG error messages rather than the default SHORT, or you might want to turn on CONFIRMation of all changes). This is most useful, of course, for those things which are not saved in the NOVA tables (primarily parameters in the SET command). In fact (until I can figure out how to change it), it should be used ONLY for SET commands. Thew reason for this is that every time you make a change to the NOVA tables (for example, defining a User command or defining a variable), NOVA pauses analysis *and does not restart it - you have to do this explicitly with an ANAL or EA command.* Thus, if you have such definitions in your NOVA_STARTUP file, analysis will *STOP* every time you enter

[a] Thanks to Corrie Kost and his group for providing this facility.

NOVA - probably not what you would like (especially if you forget to start if up again!). I think I know how to get around this, but until it is fixed, anything which alters the NOVA tables would NOT appear here (they are superfluous anyway, since the definitions are saved in the tables).

n command

Any command line which begins with a number is assumed to be a line to be inserted into the OPSEQ as line number *n*. If *command* is not present on the line (i.e. just a number is entered on the command line), NOVA enters "OPSEQ insert" mode. Multiple lines may then be entered into the OPSEQ. Use **END, EOF, EXIT** or **QUIT** to return to normal "command" mode. See the document on the Operation Sequence for a more complete description).

variable = expression

This defines the formula for calculating a variable. *Expression* can be any legal FORTRAN arithmetic or logical expression (including function calls and indexed variables). *Variable* must be one of:

i) a simple variable (no subscript)
ii) a subscripted variable which has been declared previously as a PARAMETER. In this case, *expression* may *only* be a simple constant.
iii) the name of a dimensioned variable *with no subscript.* In this case, *expression* may *only* be a call to a USRn function, and the first argument of the user function *must* be the same vector (e.g. **X = USR0 (X, ...)**). (I know this is clumsy, but you are stuck with it until I re-do the expression parser).

Examples:
X = 143.7
LOGVAR = ANGLE >= ANGLO & ANGLE <= ANGHI
Y=(X1-X2)/(X1+X2) + COS(THETA)*LOG(X12)**

Alphabetical List of NOVA Commands

ADDGROUP groupname member {member ...}

Add the items *member ...* to the group *groupname.* If groupname does not exist, it will be created - in this case the type of the group must be specified with the */TYPE* qualifier. Note that the first item in the command is the *groupname* - subsequent items are the *members* of the group.

Wild cards are not supported for the ADDGROUP command.

The command SUBGROUP is used to remove items from a group.

Qualifiers:

/TYPE=type Defines the type of the group. This qualifier is *required* when the group is first created. Legal group types are **CONDITION, VARIABLE** and **SPECTRUM**.

ANALYSE

Begin analysis. NOVA first checks to ensure that an input stream has been opened (see the OPEN command). Note that OPEN requires than an IMODE command be executed first.

NOVA then checks for a "non-executable" OPSEQ (e.g. unpaired IF - ENDIF construction). If any exist, analysis is not allowed.

NOVA then checks for any undefined entries (i.e. things which have been referenced somewhere, but NOVA doesn't yet know how to calculate them - an example would be defining X=Y+1, but forgetting to define what Y is). If any exist, the system asks if you really want to begin analysis. As far as I know, executing an OPSEQ containing undefined variables won't make the system crash, but the results will certainly be wrong. It is safest to proceed from this warning message only if you are sure that the undefined variables are in a part of the OPSEQ which will never be executed.

EA (Enable Analysis) is a synonym for ANALYSE (as is ANALYZE, for those of us who can't figure out if we are British or American).

ANALYZE

A synonym for ANALYSE.

BREAKIN

As discussed below (under the LOCK command), only one process (terminal) is allowed to change the definitions in the NOVA tables, although many terminals can view them. If you want a different terminal to be able to change things, the "owner" terminal must first execute an UNLOCK command, and then the new terminal must issue the LOCK command to gain Write access to the tables. The current "owner" or a particular NOVA subprocess is displayed on the STATus page.

There may be times when this is inconvenient (e.g. you are at a remote terminal in the proton hall, and you realize that the other guy in the counting room forgot to UNLOCK the process before going for dinner). The BREAKIN command allows you to force your way into gaining Write access to the NOVA tables. You then become the Owner of the subprocess and can change definitions (it is just like the LOCK command, except that the other guy is not required to issue an UNLOCK first).

This is potentially a very dangerous thing, since there is no indication given to the other process that this has happened (<u>it might not even be your NOVA process that you are breaking into - please be *VERY CAREFUL* when using this command - try to avoid it if at all possible</u>). In order to slow you down a little bit (and maybe make you think carefully about what you are doing), the system requires that you enter a Breakin Password (the "Password" is simply the BREAKIN command word again - no short forms allowed and it must be in UPPER CASE). The intent is not so much to prevent you from using this facility (I am assuming that people will use it responsibly, and it isn't much good having it if you can't use it) as to make you think about it before you do. A real Password may have to be implemented if this facility is abused.

Once you have done this, you will remain the Owner of this NOVA process until you UNLOCK it (or until someone else does a BREAKIN to steal it away from you - see what I mean about it being confusing and dangerous).

CLOSE

This command closes the data stream opened with the OPEN command. If the input stream is closed, then before analysis can resume, a new input stream must be opened. The subprocess must be in a *paused* condition before the CLOSE command will be accepted.

Qualifiers:

/INPUT Closes the input data stream. This is the default if no qualifier is entered,

/OUTPUT Closes the output data stream.

CONDITION name {name ...}

Define one or more variables to be of type *condition* (i.e. an integer value with an associated software scaler). Each name must be a simple variable (conditions may not be dimensioned). These may be new variables (i.e. not yet defined) or already existing variables. If they exist already, their definition (i.e. how they are calculated) will not be changed. The only difference will be that their type will be forced to INTEGER and there will be a software scaler associated with them.

CURSOR

The CURSOR command turns on the cross-hairs. They can be moved with the arrow keys on the terminal (or the mouse if you have an Atari). This command can be used if multiple plots are on the page (i.e. DSP/PAGE=n), but any commands which cause a spectrum to be redrawn (such as Expand) will redraw only a single spectrum, the one in which the cursor is currently positioned. Note that the Markers (L and R) must both be defined within the same spectrum.

The first line of the screen contains the X (and Y for 2D) coordinates of the Left marker on the left hand side of the line and the Right marker on the right hand side of the line. The second line of the screen displays different things for different functions.

The following single character commands are implemented within the CURSOR command.

A Redraw **A**ll of the spectra which were on the screen when the CURSOR command began.

Space or

C Display the **C**ounts in a particular channel (the X and Y data values associated with the centre of the channel are also given).

D **D**raw the spectrum again (but with updated data - if analysis is active, you will get a plot with the current data). If there are multiple plots on the screen, only the one pointed to by the cross-hairs will be drawn.

E **E**xpand the spectrum between the Left and Right markers.

F **F**ull display of the spectrum (expand so that all channels are displayed).

G **G**aussian fit between Left and Right markers.

L **L**eft marker for a region. Usually, the Left button of the Atari mouse is set to generate an L when you click it.

Q **Q**uit (terminate the CURSOR command and return to the NOVA CLI)

R **R**ight marker for a region (usually the Right button of the Atari mouse is set ti generate an R when you click it).

S **S**um the region between the L and R markers (gives the centroid and width as well). The region being summed is displayed as two dotted lines (for a 1D spectrum) or a dotted box (for a 2D spectrum).

W Draw the **W**indow (L and R markers). Note that if multiple spectra are on the page, there is only one Window defined (not a separate one for each spectrum) and both the L and R markers must be within the same spectrum for the window definition to be valid.

DCL {command}

If *command* is specified, a sub-process is spawned and the single DCL command is executed. If no *command* is specified, then a sub-process is spawned running VMS DCL. Commands are read from the terminal and executed as DCL commands. The terminal prompt is changed to NOVA_DCL> to remind you that you are talking to DCL from within NOVA. Use the DCL LOGOUT command to terminate the subprocess and return to NOVA.

Note that commands are executed by spawning a subprocess, which may not be possible if you have too many (VMS quotas apply). For example, if you have started a couple of NOVA subprocess and also a SCALERS subprocess to monitor scalers, you may be unable to execute DCL commands from within NOVA.

This may be combined with the User Defined Command utility to create an interface which is very much like VMS DCL. For example, to generate a directory listing from within NOVA, define

DIR := DCL DIRECTORY

and the (NOVA) command DIR behaves just like you were talking to DCL.

DEF1D /... spec1 {... spec2}

Define (or change the parameters of) one or more 1-D spectra. Note that the spectrum name comes *last* (i.e. the command is "executed" only when the spectrum name is encountered - all qualifiers before *spec1* refer to spec1 - all qualifiers before *spec2* (including those before spec1 if there are no conflicts) refer to spec2). Not all qualifiers are required in a given DEF1D command (they can be entered later), but *if the spectrum* *must be created (doesn't exist already) the /SIZE parameter must be given.* A spectrum is "undefined" until both the SIZE and DATA parameters have been specified - there are defaults for all other parameters.

Wild cards are supported (you can say DEF1D /XLOW=-500 SR* to change the lower limit of all spectra beginning with SR).

Any parameter of a spectrum *except the size* can be changed by a subsequent *DEF1D* command at any time.

Qualifiers:

/CONDITION=(list) Defines a list of 1 or more conditions in the condition list for this spectrum. *All* of these conditions must evaluate to TRUE if the spectrum is to be incremented. An empty condition list means that the spectrum will *always* be incremented. This is the default.

If only a single condition is entered, the parentheses may be omitted.

Conditions in this list will be *added* to the existing condition list for the spectrum (you don't have to re-type what is already there to add a condition). If the condition name is preceded with a '-' sign, this condition will be *removed* from the condition list. The special condition name **0** (zero) removes all conditions from the list.

Thus, to remove condition COND1 and include COND2 and COND3, specify:

/CONDITION=(-COND1, COND2, COND3)

To define the mask as COND1 only (regardless of what was there before) specify:

/CONDITION=(0, COND1)

/DATA=var (or /XDATA) Defines the data value to be histogrammed.

/LOW=var (or /XLOW) Defines the value of /XDATA associated with channel 0 of the spectrum. Defaults to 0. The value may be either a constant or the name of a variable.

/HIGH=var (or /XHIGH Defines the value of /XDATA associated with channel (SIZE-1) of the spectrum. Defaults to SIZE. The value may be either a constant or the name of a variable. For example, if you specify /XLOW=0, /XHIGH=100 and /XSIZE=100, each channel will be 1.o data units

wide. The first channel covers the range (0.0,0.999...) and the last channel covers the range (99.0, 99.999...).

Note that there is *no requirement* that (XHIGH - XLOW) be an even multiple of XSIZE! The "Bin Size" of the spectrum is (XHIGH - XLOW)/XSIZE as a *floating point number.*

/NORMAL Defines the spectrum to be a "normal" 1D spectrum (see /PTOS). This is the default.

/PTOS Defines the spectrum to be a"Parallel-to-Serial" spectrum. The data value being histogrammed is converted to type INTEGER (if necessary) and then channel 'n' of the histogram is incremented if bit 'n' of the data value is set (bit 0 is the least significant bit). Thus, this type of spectrum can be incremented several times for each event (once for each bit which is set). A /SIZE parameter greater than 32 is meaningless (although allowed) as there are only 32 bits in an INTEGER*4 value. The Underflow channel will be incremented if the value is zero. The Overflow channel will never be incremented.

/SIZE=n (or /XSIZE) Define the number of channels in the spectrum (>0, INTEGER constant). This qualifier is *required* when the spectrum is first defined.

Examples:
DEF1D /SIZE=100 SX
DEF1D /XDATA=X /XLOW=-100 /XHIGH=200 SX
DEF1D /CONDITION=0 SX

DEF2D /... spec1 {/... spec2}

Define one or more 2D spectra. See the comments for *DEF1D* above - the *DEF2D* command is similar. Qualifiers *must* include either an X or Y in them (/SIZE is no good - it must be either /XSIZE or /YSIZE). When the spectrum is first defined, *both /XSIZE and /YSIZE must be given.* Note that the maximum size for a 2D spectrum is approximately 100 x 100 channels (this limit is actually imposed by the DSP command - larger 2D spectra can be defined and incremented OK but they can't be displayed properly by NOVA. You have to use PLOTDATA or some other graphics package to display them). The system will issue a warning if you try to define a spectrum that is too big, but it will do it for you anyway. If you want to change it, you must first DELETE it and then re-define it because I can't change the number of channels (yet).

Qualifiers:
/CONDITION=(list)

/NORMAL

/PTOS (if specified, both X and Y axes are assumed PTOS-type variables - there is no provision for one PTOS and one NORMAL coordinate).

/XSIZE=nx

/YSIZE=ny

/XDATA=var

/XLOW=var

/XHIGH=var

/YDATA=var

/YLOW=var

/YHIGH=var

Examples:
DEF2D /XS=50/YS=50/CON=C1 SXY
DEF2D /XD=X /YD=Y /CON=(-C1, C2) SXY

DELETE name {name ...}

Delete one or more items. Entries will be physically deleted (if possible) or marked as Undefined.

Wild cards are supported by the DELETE command.

System variables (i.e. things that NOVA defines for you, such as **$CONDMASK**) are protected. Thus, the command
DELETE *

deletes all user variables but *not* system defined variables.

It is not possible to delete something which is referred to by something else. For example, if you have defined

X=Y+1

you can't delete Y because NOVA realizes that it is required for the calculation of X (you could delete Y if you deleted X first). If you try, NOVA will complain that the variable is "in use" and therefore can't be deleted. Its definition will be removed, however, and it will appear as an Undefined variable (i.e. as if you had just entered the above definition for X but not yet defined Y).

Qualifiers:
/CONFIRM Items to be deleted are listed on the terminal. The user must then enter one of the single letter commands:
Y Yes - delete this item
N No - keep this item

A All - delete all the rest (i.e. just like **Y** for all remaining items in the list)

Q Quit this bunch

/ORPHAN Only those items which are "orphans" will be considered as candidates for deletion (an *orphan* is an item which is not referred to by anyone else, and hence could be deleted without changing things). If no *names* are entered, the default for /ORPHAN is * (i.e. just DELETE/ORPHAN will *delete all entries which are orphans*).

/VERIFY Names of all deleted items will be echoed on the console.

Examples:
DELETE X1, Y1, Z*
DELETE/CONF *
DELETE/ORPHAN

DELETE/OPSEQ n1 {n2}
DELETE/OPSEQ ALL

Delete lines from the OPSEQ. This statement is (somewhat) obsolete now that the full screen editor for the OPSEQ has been implemented.

If only n1 is given a single line will be deleted. If both n1 and n2 are given, lines n1 to n2 (inclusive) will be deleted. If ALL is entered, the entire OPSEQ will be deleted.

The first line (**$BEG_OPSEQ: CONTINUE)** and the last line **($END_OPSEQ: CONTINUE)** of the OPSEQ are protected, and cannot be deleted. Thus a command like

DELETE/OPSEQ 10 999999

would delete all lines from 10 through to the end of the OPSEQ (but not $END_OPSEQ).

Qualifiers:
/CONFIRM

/VERIFY As for the DELETE command above.

DIMENSION var(max) {var (max) ...}

Define one or more variables as *vectors* (things requiring an index). *Max* is the maximum index allowed for the variable - <u>*the minimum index is always 0*</u> (and hence the number of values is really max+1). The variables must not have been dimensioned before (you can only do this once - I can't (yet) change the size of a dimensioned variable) and it must not have appeared before without an index (since this implicitly defines it to be a simple, undimensioned variable, and I can't change that yet either). Variables can also be dimensioned in type declarations statements (e.g. INTEGER*4 X(40)).

DSP /... name1 {, /... name2 ...}

Displays one or more spectra on the terminal. The spectrum name must come *last* on the command line (that is, the command string is scanned left to right, but only "executed" when a spectrum name is encountered). Thus, all qualifiers preceding *name1* affect the display of *name1*, all qualifiers preceding *name2* (including those which precede *name1* where there are no conflicts) affect the display of *name2*, etc.

Each *name* may be either a single spectrum name, or the name of a spectrum *group*. If it is a spectrum group, the effect is as if all members of the spectrum group (modified by any preceding /OFFSET qualifier, if present) were entered instead of the spectrum group name. Wild cards are supported for the DSP command.

Multiple spectra may be displayed on one page.

Qualifiers:

/ATARI Draw the spectra using the "fast histogram" option of the Atari ST640 package. This is the default if the terminal type has been set to Atari using the SET command. It is much faster than the /VT640 option, but (of course) requires that you are using an Atari 1040 running ST640.

/COLUMN=n Display a column (fixed X data value) of a 2D spectrum as a 1D spectrum. n is the X data value of the column to be displayed.

/FULL Normally, the first and last channels of a 1D spectrum (or the outer "box" around a 2D spectrum) is ignored when displaying the spectrum. The /FULL option forces these channels to be included.

/HIGH A synonym for /XHIGH when displaying a 1D spectrum.

/LOW A synonym for /XLOW when displaying a 1D spectrum.

/MAX=n Force the maximum scale to be *n* counts. All channels with counts >= n will be displayed as "full scale".

/OFFSET=n When the name of a *spectrum group* is encountered in the command line, the effect is as if all of the members of the spectrum group were entered in place of the group name. The /OFFSET qualifier modifies this by starting with member number *n* of the group. For example, if *SPECGROUP* is a spectrum group containing 12 spectra, the command

DSP /OFFSET=9 SPECGROUP

would display the last four spectra (numbers 9, 10, 11 and 12) in the group.

/PAGE=n Format the display page so that a maximum of *n* spectra can be drawn on the page. Values are effectively rounded up to the next larger number of the series (1,2,4,6,9,12,16,20,25). If this option is not used, the format will be chosen based on the number of

spectrum *names* in the command line. Note that (for example), if /PAGE=3 is specified, the display page will be formatted for 4 plots (the next larger number) but only 3 plots will be produced.

Numbers larger than 9 produce plots on which very little detail is visible.

/ROW=n Display a row (fixed Y data value) of a 2D spectrum as a 1D spectrum. n is the Y data value corresponding to the row to be displayed.

/VT640 Draw the plots using the TRIUMF standard graphics driver for a VT640 terminal. This is the default (unless you use the SET TERMINAL command to modify it). It can be used if the terminal you are using is an Atari, but it is much slower than the */ATARI* OPTION.

/XHIGH=n The largest channel displayed in the X direction will be that corresponding to the value XDATA = n. The synonym /HIGH may be used for a 1D spectrum. If not entered, the default is the last channel of "real data" (i.e. excluding the overflow channel) unless the /FULL qualifier is used.

/XLOW=n The smallest channel displayed in the X direction will be that corresponding to the value XDATA = n. The synonym /LOW may be used for a 1D spectrum. If not entered, the default is the first channel of "real data" (i.e. excluding the underflow channel) unless the /FULL qualifier is used.

/XPROJ Displays an X projection of a 2D spectrum as a 1D plot. For each X channel, all Y channels between /YLOW and /YHIGH will be summed. The underflow and overflow channels will not be included unless the /FULL qualifier is entered.

/YHIGH=n The largest channel displayed in the Y direction will be that corresponding to the value YDATA = n. If not entered, the default is the last channel of "real data" (i.e. excluding the overflow channel) unless the /FULL qualifier is used.

/YLOW=n The smallest channel displayed in the Y direction will be that corresponding to the value YDATA = n.If not entered, the default is the last channel of If not entered, the default is the first channel of "real data" (i.e. excluding the underflow channel) unless the /FULL qualifier is used.

/YPROJ Displays a Y projection of a 2D spectrum as a 1D plot. For each Y channel, all X channels between /XLOW and /XHIGH will be summed. The underflow and overflow channels will not be included unless the /FULL qualifier is entered.

DUMP filename

Dumps the NOVA tables and/or spectra to a disk file. The LOAD command is used to retrieve the information at a later time.

The *DUMP* command checks for undefined entries, and warns you if any are present before it creates the dump file.

Qualifiers:

/ALL Dump both the definitions and the spectrum contents to *filename.* This is the default (except for /ASCII)

/ASCII Dump the definitions only (no spectrum contents) as an ASCII (i.e. human-readable) file. This file can be edited using a standard text editor. If no extension is given, the default .ASC is assumed.

/BINARY Dump the information in Binary (i.e. computer- readable). This is the default. If no file extension is given, the default .BIN is assumed.

/SPECTRA Dump *only* the contents of the spectra (no definitions). It is up to the user to ensure that, when this file is later LOADed, the table definitions match the one that was active when the spectra were dumped. Use of this option is *not recommended* until I can think of a way to protect you against table / spectrum incompatibilities.

/TABLES Dump only the tables (i.e. definitions). This is the default (in fact, all that is allowed) for /ASCII.

ECHO {string}

If *string* is present in the command line, then this command will simply type it on the terminal.

If *string* is not given (i.e. just the command word *ECHO* is entered), then all further commands read from command files will be listed on the terminal before they are executed. Use the command *NOECHO* (or SET NOECHO) to disable this.

EDIT name

This command list the FORTRAN expression which was used to define the variable *name*, and allows you to edit it using the cursor keys on the terminal (similar to the VMS RECALL command, except that you are recalling the definition of a variable rather than a recent command line). Note that only *variables / conditions* can be edited (it is not needed for spectrum definitions - you can just enter the new definition of a parameter in a spectrum definition and it will supersede the old one). You cannot use this facility if you have SET TERMINAL /REMOTE, as the command recall function is not available in this mode.

EDIT/OPSEQ

This command invokes a full-screen editor (the VAX-VMS EDT editor) to edit the OPSEQ. The first line ($BEG_OPSEQ: CONTINUE) and the last line ($END_OPSEQ: CONTINUE) will not be included (since you had better not delete them, so I just don't give you the option).

The editor works just like EDT (in fact, it *IS* EDT with the exception that no .JOUrnal file is produced. In particular use EXIT to exit the editor and save the changes that you have made in the OPSEQ, and use QUIT to exit with no changes (i.e. abort and leave the OPSEQ as it was).

If you have a favourite editor command file to customize EDT, define the Logical Name EDTINI before you start NOVA.

$ DEFINE EDTINI yourstartupfile

END

This is a synonym for the EXIT command.

EOF

This is a synonym for the EXIT command.

ERASE

Erases the text, graphics or both screens on your terminal.

QUALIFIERS:

/ALL Erases both the text and the graphics screen. This is the default.

/GRAPHICS Erases only the graphics screen. The text screen is unaffected.

/TEXT Erases only the text screen. The graphics screen is unaffected.

EXIT

The EXIT command is used primarily to terminate "OPSEQ insert" mode (i.e. stop entering lines into the OPSEQ) and return to normal command mode. END, EOF, QUIT and CTRL-Z are synonyms for EXIT.

If entered from the console in command mode (i.e. when you are not entering lines into the OPSEQ), any of these commands terminates the NOVA program and returns you to VMS DCL.

Qualifiers:

/SAVE When you exit NOVA "normally" (e.g. with the EXIT command rather than CTRL-Y), the backup file (NOVA_BACKUP.TBL) is deleted - the system assumes that you are happy with everything that you did and don't wish to retain the older version. If you use the /SAVE qualifier, the backup file will be retained. If you wish to keep the file

permanently, you must rename it to prevent the system from deleting it the next time you exit from NOVA normally.

GCOLL

Force "Garbage collection" to occur. All items in NOVA are stored in one large table. As things are deleted (either explicitly or internally by NOVA) "holes" will be left in the tables. This command reorganizes the tables to eliminate these holes, and also reports how much of the table is used.

This command is almost obsolete, since garbage collection is now automatic after every NOVA command is executed. It does, however, do a (minimal) check on the table structure to ensure that things haven't become corrupted.

HARDCOPY

Produces a hard copy output of the graphics screen (i.e. what you did most recently with the DSP command). Two modes are supported:

1. If you have SET /PRINTER=REMOTE (or used the /REMOTE qualifier in the HARDCOPY command), the plot will be printed on the VAX printer queue HP$LASER (you must have defined this logical name before entering NOVA). This is supported *only if the DSP command used the /VT640 option* (i.e. if the plot was produced using the TRIUMF Graphics package)

2. If you have SET /PRINTER=HPLASER or EPSON (or used the /HPLASER of /EPSON qualifier in the HARDCOPY command), the assumption is that the printer is a Local Printer (i.e. attached directly to an Atari 1040).

HELP {topic {subtopic ...}}

If no *topic* is given, the system lists the topics available. This is very similar to the VMS HELP command. Help on NOVA may be obtained from VMS DCL by using the DCL command:

HELP/LIBRARY=NOVA$DIR:NOVAHELP

IMODE mode

This selects the format of the input data expected by the analysis subprocess.

NOVA can analyse data written by many different acquisition programs, but it must be told what format the data is in. The formats currently recognized are:

LNS SARI data acquisition program at Saturne. This is currently supported only for a particular experiment (Expt. 177 at LNS), but could easily be modified to handle other formats should the need arise.

MIDAS The MIDAS data acquisition system used at TRIUMF for the Charge Symmetry Breaking experiment (and a few others).

NOVA Data generated by the NOVA program itself (using the OUTPUT command in the OPSEQ). NOVA output tapes are identical to VDACS tapes, so either IMODE NOVA or IMODE VDACS may be used.

ONLINE This format allows analysis of events generated by an online data acquisition process running in a Starburst J11.

TEST This generates pretend data. The event type is always 1, the length is 100, and $RAW(I) = I (i.e. the value of each raw data word is equal to its index in the raw data vector).

VDACS This format allows analysis of data tapes written by the TRIUMF online acquisition system.

It is a relatively simple job to include other data formats into NOVA (although probably not one that I would like to allow the unwashed user to tackle). Output from a Monte Carlo code is something which is conspicuous by its absence from the above list. If you would like NOVA to be able to analyse data in a different format, let me know and I will write an input driver for your particular input format.

*INTEGER*2 name {(max)}, {name ...}*

Declares one or more variables to be of type INTEGER*2. Variables may be dimensioned by this command also (see the DIMENSION command above). The "*2" is required (since the default for Integer variables is INTEGER*4), but the keyword INTEGER may be abbreviated (INT*2, I*2, etc.).

Examples:
INTEGER*2 X, Y, Z
INT*2 I2VECTOR (100)
I*2 I2VALUE

*INTEGER*4 name {(max)} {name ...}*

Declares one or more variables to be of type INTEGER*4. Variables may be dimensioned by this command also (see the DIMENSION command above). The '*4' part is optional, since Integer variables are INTEGER*4 by default.

Examples:
INTEGER*4 I4VECTOR (20)
INT I4VALUE

LIST name {name ...}

Lists the current definition of one or more items. Wild card characters in *name* are supported. Information listed depends on the type of the entry and on the complexity of listing requested. If no *name* is entered, *all* items will be listed (i.e. the default is **LIST** *).

If *name* contains wild cards, the items will be listed in *alphabetical order,* not in the order in which they were defined.

Qualifiers:

/1D List only items of type 1D SPECTRUM.

/2D List only items of type 2D SPECTRUM.

/ALL Normally the listing does not include system-defined variables (**$ALLCONDS, $RAW,** etc.) The **/ALL** qualifier forces these items to be included.

/BRIEF Only the name is listed (5 per line).

/CMASK List the conditions which are currently part of the system condition mask. This will include all conditions mentioned in **/CONDITION=(list)** for all spectra. Currently the maximum number allowed is 128.

/CONDITION List only items of type CONDITION.

/CONSTANT List only items of type CONSTANT (either scalar constants defined implicitly - X=143.2 for example), or names which have been explicitly declared using the CONSTANT command.

/DECIMAL List all integer values in decimal. This is the default.

/FULL A complete listing (usually 2 or 3 lines) of everything known about the item is produced. This is the default if *name* contains no wild cards. For vectors (i.e. things which are dimensioned) the /FULL qualifier also lists the values of the entire vector.

/GROUP List the *members* of any item which is a group. Normally, only the name of the group and the number of members in the group will be given.

/HEXADECIMAL Lists all Integer values in hex.

/I2 List only items of type INTEGER*2.

/I4 List only items of type INTEGER*4.

/LABEL List only items of type OPSEQ LABEL.

/NORMAL A one line description of the most commonly needed information is produced. This is the default when *name* contains at least one wild card character.

/OCTAL List all Integer values in Octal.

/ORPHAN Lists only items which are *orphans.* See the DELETE command for a discussion of "orphans".

/REAL List only items of type REAL*4.

/REF Lists *references to* each item rather than the definition of the item. The default here is */BRIEF* (i.e. list only the names of things which refer to a given item). This is useful when the system won't let you delete something, to find out who is still referring to it.

/SPECTRUM List only items of type SPECTRUM. This is equivalent to LIST/1D/2D.

/UNDEFINED Lists only those items which are undefined. When you are all done defining things, there must be no undefined entries left or the system will not let you begin analysis (or at least will ask if you are sure that you know what you are doing).

/USER Lists only those items which are of type User Defined Command.

/VARIABLE Lists only items of type VARIABLE. This is equivalent to LIST/I2/I4/REAL.

Many of these qualifiers may be combined. for example:

LIST/UNDEFINED/1D lists only 1D spectra that are undefined.
LIST/REAL/I4 lists only REAL and INTEGER*4 variables.

/BRIEF, /NORMAL and /FULL may be used with any other qualifiers.

A word of WARNING!

The LIST command can take a long time complete (long on the time scale of analysing events, at least), and *analysis continues in parallel with it!* If you are LISTing things which are calculated by the OPSEQ (as you probably are), you might get values for different variables which *come from different events.* I think I know who to fix this, but until I do, it is safest to always PAUSE or DA before doing a list of calculated variables.

Examples:
LIST/VAR X* List all variables starting with X
LIST List everything except system variables
LIST/BR/S1D List names only of all 1D spectra

LIST/IF n1 n2

List only the IF statements in the OPSEQ between lines n1 and n2 (inclusive). If n1 and n2 are omitted, all IF statements in the OPSEQ will be listed. This listing differs from the LIST/OPSEQ (below) in that only the IF statements in the OPSEQ are listed. The software scalers for each IF statement are also listed (how often the statement was executed and how often the *expression* was TRUE) unless the /NORMAL qualifier is also used.

Qualifiers:

/NORMAL List only the IF statements but not the software scalers. The default is to list the values of the software scalers as well.

LIST/OPSEQ n1 n2

List lines n1 through n2 (inclusive) of the OPSEQ. If n2 is omitted, only line n1 will be listed. If both n1 and n2 are omitted, the entire OPSEQ will be listed.

Qualifiers:

/FULL With each IF statement, list the values of the software scalers for the IF statement. The default is to list the statements only.

LOAD filename

This command loads tables and / or spectra from a file created with the DUMP command. *All previous information in the tables is overwritten.*

Dump files may be either in Binary or ASCII (see the DUMP command). LOAD is clever enough to figure out what the format of the file is - you don't have to tell it.

For an ASCII file, you can either *LOAD* the file, or use *@filename* to execute the commands in it as if they were entered from the keyboard. The difference is that *@filename* will <u>*add*</u> the definitions to any existing ones, while *LOAD filename* will delete everything that is there first (i.e. it will overwrite the existing tables).

Eventually, I hope that you will be able to load only certain definitions from a dump file (e.g. only certain spectra, or values of certain parameters). Such selective LOADs are not implemented yet.

Earlier versions of DUMP files are automatically converted to the most recent version when they are LOADed (you will see a descriptive message when the LOAD command tries to perform this conversion). If problems occur, please save the offending files so that I can fix the problem.

Qualifiers:

/ALL Load both table definitions and spectra (if present) from the dump file. This is the default.

/SPECTRA Load only the spectrum data but not the table definitions. It is the user's responsibility to ensure that the spectrum data corresponds to the table structure that he currently has. Use of this qualifier is not recommended.

/TABLES Load only the table definitions but not the spectra.

LOCK

Only one process (terminal) is allowed to have *write access* to a particular NOVA subprocess (i.e. is allowed to change definitions in the tables) at any time, although any number of users can look at things. By default, the user who starts the NOVA subprocess (with the NOVASTART command) has exclusive write access. The STATUS command indicates who currently has write access to a particular subprocess.

In certain instances, it might be nice to allow another user write access to the tables (for example, you might have a remote terminal close to the electronics set up). If the first (initiating) process issues an *UNLOCK* command, a second process can the *LOCK* the NOVA subprocess and thereby gain write access to the tables. Note that the current owner process must first *UNLOCK* the tables. (See, however, the *BREAKIN* command).

If you attempt to alter the tables of a process you haven't locked, NOVA will issue the error message *No write access.*

MAPTO {name}

It is possible to have several NOVA subprocesses running at once, limited only by the constraints imposed by VMS (e.g. subprocess quota). Only one of them is "active" at a given time, however (that is, they can all be busy analysing data, but the user sitting at a terminal can only communicate with one of them at a time). This command allows you to change which NOVA subprocess you are talking to.

If *name* is given, the system will MAPTO the subprocess named *NOVA_name.* If *name* is not entered, the system will list all of the NOVA subprocesses which belong to you, and you may then select which one you want.

When you begin a new subprocess (with the VMS NOVASTART command), it becomes the "active" subprocess by default, so it is not necessary to explicitly MAPTO it.

When you MAPTO a subprocess, it remains the active one until you explicitly change it (*even if you exit NOVA to DCL and then re-enter NOVA, so be careful*).

NEW

Erase *everything* in the tables. The system will ask you for confirmation unless the /OK qualifier is entered.

Qualifiers:

/OK Do not require keyboard confirmation of the deletion (i.e. just go ahead and do it).

NOECHO

Turns off the automatic echoing of commands in @-files (see the *ECHO* command). You can also use the SET NOECHO command.

OPEN/INPUT {file/device}

Opens a file / device in preparation for analysis. The /INPUT qualifier is optional (i.e. the default is /INPUT). Depending on the input mode / format (set with the IMODE command), *filename* may not be required.

The input mode must be set before you enter the *OPEN* command. If a file was open previously, it must first be *CLOSEd* before another one can be opened.

If the input mode is *ONLINE*, the system will attempt to open the default channel GRA1: unless the user requests a different channel in the OPEN command. If the input mode is TEST, no filename is permitted. In all other cases (where you are presumably analysing old data) the *file* parameter specifies the device (i.e. mag tape) or disk file to open.

OPEN/OUTPUT filename

Opens a file / device to write events produced by the OPSEQ OUTPUT command. The /OUTPUT qualifier is required (the default for OPEN is /INPUT). Filename is either a tape unit or a disk file.

Opening an output file effectively "enables" output from the OPSEQ (i.e. if you don't want to produce output any more, the only way to do it is to CLOSE the output channel).

Note that this output device is <u>not the same as the one that the online VDACS is using to log raw event data.</u> It contains only analyzed events, and is intended to be used for event skimming (for example), not data logging.

PARAMETER name {name ...}

This defines one or more names to be PARAMETERS (i.e. things which NOVA doesn't have to try to calculate before it uses them). This statement is most often used for dimensioned variables, since simple (scalar) parameters are implicitly defined as such by simple entering

name = value

Vectors *must* be declared PARAMETER before you can set their values with simple assignment statements (as in **X(n) = value**).

These are not really PARAMETERS in the FORTRAN sense of the word - Constants would be a better term (in fact, in earlier versions of NOVA they were called CONSTANTS). I prefer the term PARAMETER mostly because Constant is too close to Condition (in older versions of NOVA, you couldn't say LIST/CON because CON could mean either CONSTANT or CONDITION).

PAUSE

Stops analysis by the subprocess. Analysis will stop at the end of the current event, and will remain stopped until another *ANALYSE* OF *EA* command is entered. *DA* (Disable Analysis) may also be used.

It is currently not possible to alter definitions while the subprocess is actively analysing data. Thus, if you attempt to change a definition, NOVA will automatically *PAUSE* the subprocess for you (it will warn you that it has done so) *and it will remain paused until you start it up again*. As discussed above, this is one very good reason for not putting definitions of User commands in your NOVA_STARTUP file.

QUIT

QUIT is a synonym for END, EOF or EXIT. It is used primarily to terminate OPSEQ insert mode. If entered in COMMAND mode (i.e. while you are talking to the NOVA Command Line Interpreter), you will be returned to VMS DCL.

*REAL*4 name {(max)} {name}*

This declares one or more variables to be of type REAL*4. Variables may be dimensioned by this command also (see the DIMENSION command above). The '*4' part is optional (since only REAL*4 variables are implemented).

Examples:
REAL*4 XVECT (20)
REAL X,Y

SET

Sets various parameters defining the operating environment for the system. *Note that these parameters are not saved from one session to another*, and hence must be re-entered every time you type NOVA (or perhaps entered in your NOVA_STARTUP file, where they will be executed automatically every time you start a NOVA session).

Qualifiers:

/CONFIRM=option Defines under what conditions NOVA will request confirmation before changing definition. The options are:

- **ALL** Equivalent to both CHANGE and NEW.
- **CHANGE** Confirmation is required if you attempt to *change* the definition of an existing variable.
- **NEW** Confirmation is required if you attempt to define a new variable / spectrum.
- **NONE** (Default). No confirmation is required.

/ERROR=option Defines the format of error messages produced by NOVA.

- **SHORT** (Default) produces a cryptic (usually 1 or 2 words) error message.
- **LONG** produces a longer (1 or 2 lines, usually) error message. SET /ERROR=LONG also will echo the most recent error message in its long format (so that if the short one is too cryptic, you can find out what it really means)

/INPUT=option Defines how input lines are read from your terminal.

LOCAL (Default). Input is read 1 character at a time. This mode is required for such things as command recall / editing using the cursor keys.

REMOTE Reads input 1 *line* at a time. It is less convenient, since command recall / editing is not available. However, when logging in over a network from a remote site (e.g. DATAPAC or CANET), single character I/O can be *extremely slow* (any of you who have tried to use the EDT editor over the network will know what I mean). You can, of course, switch from one to the other at will (i.e. you can SET /INPUT=option whenever you want, and may switch between the two modes at any time).

/MAYPROCESS (Default). VDACS (IMODE = ONLINE only) will pass events to NOVA for analysis only if there is time (i.e. NOVA will be kept as busy as possible, but acquisition will not be slowed down just because events are not being analysed quickly enough). Note that event types which are marked MUSTPROCESS in your TWOTRAN program will all be analysed.

/MUSTPROCESS VDACS (IMODE = ONLINE only) will ensure that all events generated by the acquisition program will be passed to NOVA for analysis (slowing down acquisition if necessary). Note that this could be *dangerous*, since if NOVA cannot keep up, acquisition will be slowed down (and VDACS sometimes gets annoyed if it runs out of data buffers).

/PRINTER=type Defines the type of printer you are connected to (so the *HARDCOPY* command knows how to generate a hardcopy of the screen). Current options are:

EPSON Locally connected to an Atari)

HPLASER (default - Locally connected to an Atari).

REMOTE Plots will be generated on the VAX queue HP$LASER.

/PROCESS=option This is an alternative format for specifying MAYPROCESS or MUSTPROCESS. The options are:

MAY (default)

MUST

/SNAP=option This controls the action of the CURSOR command when you select a channel. The options are:

OFF The crosshairs will be left where they are

ON (default) The crosshairs will "snap" to the centre of the selected channel.

/TERMINAL=type Defines the type of terminal you are sitting at. The options are:

ATARI

VT640 (default). This option is currently used only for drawing histogram displays. If the terminal type has been set to

ATARI, the system will use the ST640 fast histogramming package to draw the plots (this can be over-ridden in the *DSP* command by explicitly specifying /VT640). If the terminal type is set to VT640, then the system will use the VT640 driver in the TRIUMF graphics package to draw the plots. Thus, if your terminal really is an Atari, you could still specify **/TERMINAL=VT640** if you want the plots to be drawn without using the ST640 fast display package).

If you want to generate hardcopy on the VAX printer queue HP$LASER (SET /PRINTER=REMOTE), you must generate the plots using the /VT640 option.

SHOW

This lists the current values of the options defined using the SET command.

STATUS

Shows you the status of the analysis subprocess (number of spectra defined, number of events analysed, etc.). This page also shows the current "owner" of the subprocess (i.e. the process which is allowed to change things) or the keyword AVAILABLE if no one currently owns it. The display updates every 0.5 seconds. Hit any key on the keyboard to terminate this command (note that the character you type is "gobbled up" by the STATUS command).

SUBBROUP groupname member {member ...}

Remove members from the group *groupname.* This is the inverse of the ADDGROUP command. If an item is part of a group (e.g. a spectrum), it must be removed from the group using the SUBGROUP command before it can be deleted.

Wild cards are supported in the SUBGROUP command.

SWRITE specname filename

Writes the contents of the spectrum *specname* into the disk file *filename* in ASCII (i.e. human readable form).

The files are human readable, but the format is not necessarily obvious. The following information is contained in the file. Note that (for 2D spectra), "X" refers to the horizontal axis and "Y" refers to the vertical axis.

(Apologies - it is much easier to explain this if I switch into "computerese" for a bit).

begin {computerese}

There are four kinds or "records" in the file.

TITLE Contains the type of spectrum (1 = 1D, 2=2D) and the name of the spectrum.

FORMAT (I2, 1X, A12)

AXIS Contains the number of channels along the axis, the lower and upper limits, and the name of the variable.
FORMAT (I6, E16.7, E16.7, A12)

STAT Contains "statistics" for a given row (or column for 2D) of the histogram. Sum(X), sum(x**2), minval, maxval (minval is the minimum value you tried to histogram, maxval is the maximum value you tried to histogram. If all of your counts appear in the underflow or overflow channels, these values will tell you what your histogram limits should be to see the data).
FORMAT (D16.7, D16.7, E16.7, E16.7)

DATA Contains the data for a given horizontal (X) row of the spectrum. There will be (#X channels + 2) values, written 5 per line (thus, there may be several lines in this "record", and the last might not have all 5 values in it. If the number of X channels is 12, there will be 2 lines with 5 values and 1 line with 4 values). The first value is the underflow channel, the next (#X channels) values are the "real data", and the last value is the overflow channel.
FORMAT (5E16.7)

The files produced by SWRITE look like:

1D **TITLE** record

AXIS record for X axis
STAT record for 1D spectrum
DATA record for 1D spectrum

2D **TITLE** record
AXIS record for X axis
AXIS record for Y axis
STAT records for the vertical (Y) columns of the spectrum. There will be (#X channels + 2) of these records. The first is for the X underflows, then (#X channels) of "real data", and the last is for the X overflows.
STAT records for the horizontal (X) rows of the spectrum. There will be (#Y channels + 2) of these records.
DATA records. There will be (#Y channels + 2) of these. The first is Y underflows, then (#Y channels) or "real data", and the last is for the Y overflows. Remember that each of these "records" might contain several lines of 5 numbers each, as described above.

end {computerese}

TITLE {string}

If string is given, this command sets the TITLE to be this string (80 characters maximum). If string is absent, the current title will be listed on the terminal.

This title is purely descriptive, and has no function in the NOVA system.

UPDATE

Makes a new backup file (NOVA_BACKUP.TBL) containing the current state of the NOVA tables.

Every time you run the NOVA program to change definitions in the tables, the system creates a backup file of the state of the tables, so that if you really mess things up, you can CTRL-Y to abort and then restore the state of the system from the backup file (the backup file is created only when you actually *change* something, so that commands which don't alter the tables execute more quickly - this is why the first command that you enter in NOVA might take several seconds to execute). If you wish, you can use the *UPDATE* command to create a new backup file at any time. This is just like doing a *DUMP/TABLES*, except that the file name is automatically set to NOVA_BACKUP.TBL.

ZALL

Clears all spectrum counts, and zeroes all software scalers for all conditions, spectra and OPSEQ IF statements.

ZERO name {, name ...}

Zeroes one or more spectra or conditions. Zeroing a condition clears the software scalers for the condition. Zeroing a spectrum clears both the software scalers and the counts in the spectrum.

Zeroing a vector which has been declared as a PARAMETER clears all elements of the vector (e.g. ZERO SCALERS - see the discussion of SCALERS processing at the end of the NOVA OPSEQ manual). Note that ZALL does NOT clear SCALERS (or any other PARAMETERs).

You cannot clear the software scalers for an OPSEQ IF statement using the ZERO command - you must use ZALL for this.

Wild cards are supported for the ZERO command. The command *ZERO ** would clear all spectra and condition counts, but is much slower than the *ZALL* command (however, ZALL also clears the condition counts for the OPSEQ IF statements).