NOVA (Version 2.3)

This document presents a summary of the NEW features of NOVA implemented since the last documentation (labeled Version 2.0). It is intended to act as an update to (rather than replacement for) the previous document. Also , this documentation was produced long after some of the enhancements were implemented, and since I have been very lazy, I didn't document everything I did while I was doing it, so it is possible (indeed likely) that something got left out. If you find something which is missing, please let me know.

Additionally, there are some things in here that are not really new - they have been around for a while, and just never documented properly.

ASCII variables

NOVA now recognizes variables of type ASCII (which contain an arbitrary sequence of ASCII characters of any length). They are particularly useful in conjunction with the BNK2 and BNK4 functions described below for locating data banks within events. The syntax for defining them is similar to other variable types in NOVA

```
ASCII JUNK
JUNK="this can be any ASCII string"
```

The double quotes around the value may also be single quotes (and in fact can be omitted completely).

Batch Jobs

There was a long-standing problem when using NOVA in a Unix batch environment - commands like 'novastart', 'novakill' and 'nova' didn't work. You instead had to say things like $NOVADIR/nova.exe. Passing arguments to these programs was also sometimes problematic.

This has been fixed now, but (sorry) only for those of you who use bash as your shell. Commands like 'novastart' and 'nova' whould work fine in shell scripts. If you use tcsh, you are stuck with the old syntax.

BNK2 (ASCII)

The BNK2 function locates a data bank (by name) in the $RAW data vector, and returns a pointer (index) to the first data variable in the bank. The argument is an ASCII variable (4 characters - case sensitive) containing the name of the data bank.

```
ASCII CBNK
CBNK="TDC0"
```

```
IBNK = BNK2 (CBNK)
```

IBNK will contain an index into $RAW of the first data value in the bank named TDC0. For example, you could use this index as:

```
ID0=$RAW(IBNK)
ID1=$RAW(IBNK+1)
```

to access successive (INTEGER*2) data values in the bank.

This function (and BNK4 described below) are useful only for events in MIDAS/YBOS formats (since only these formats support data banks).

BNK4 (ASCII)

This function is identical to BNK2, except that it returns an index into the INTEGER*4 raw data vector $RAW4.

```
ASCII CBNK
CBNK="TDC0"
IBNK = BNK4 (CBNK)
ID0 = $RAW4(IBNK)
ID1=$RAW4(IBNK+1)
```

ID0, ID1 will contain the first two INTEGER*4 data values in the bank named TDC0.

BNK4 can also be used to find/access banks in $RAWFP.

Command History

The command history (i.e. List of most recently executed commands) is saved between NOVA sessions (i.e. If you exit NOVA and then come back in, the command history is retained). In addition, the number of saved commands has been increased to 41 (from 21).

If you have an old history file lying around (this capability has actually been present for a while - one of the things I was meaning to document and never got around to), you might get an error the first time - it will say "Command History not Restored". Just ignore the error - it should work OK the next time. If not, delete the file NOVAHSTY.

See also SET HISTORY

Compressed DUMP Files

Dump files produced by the NOVA DUMP command now incorporate (minimal, but fairly effective) compression algorithms (primarily suppression of zeroes). As a result, the files produced by the DUMP command can be significantly (up to a factor of 3 to 4) smaller than previously. To remind you of this, the default extension has been changed from .nbi (Nova BInary) to .nbc (Nova Binary Compressed).

Three levels of compression are provided. See SET FILECOMP.

DEBUG n

This command sets a flag which results in descriptive messages being printed to the screen at various points throughout Event Analysis. It is a bit mask - different bits result in different messages.

This is not so useful for the average user (mostly it exists for my convenience in tracking down problems), but could be useful if your analysis code gets "stuck" (or crashes) to tell you what the code was trying to do when it died.

DEBUG 0 (the default) turns off these messages.

DESCRIBE variable {text}

This command allows you to enter and/or view a DESCRIPTION of a variable. The description is not used by NOVA - it is intended solely for humans, and allows you to document what a variable is. If 'text' is not entered, the current description is printed (the description is also given with the LIST/FULL command).

```
DESCRIBE xfk Focal Plane position in mm
DESCRIBE xfk
xfk Focal Plane position in mm
```

DSP/COMP

The DSP command now allows you to sum together adjacent channels before displaying a spectrum, allowing you to more easily see relevant features in spectra which have many channels and/or only a few counts in each channel. The syntax

```
DSP/COMP=n specname
```

will sum n adjacent channels of the spectrum before display (in both X and Y if it is a 2D spectrum).

DSP/LOG

For X-windows display only (SET /TERM=X, not SET /TERM=DECW), a log plot has been implemented.

DUMP

The DUMP (and also LOAD) commands have been modified to use the environment variable NOVADUMPDIR. See the discussion of this variable below for details.

Environment Variables

The following environment variables might be useful (they are described in detail individually)

NOVADATADIR	Directory containing Raw Data files
NOVADUMPDIR	Directory containing dump files
NOVAMAIN	Location of novamain.exe program
NOVAPRINT	String used to make hardcopies

ERUN

The ERUN command causes the (User-supplied) subroutine USRERUN to be called (in novamain.exe) , passing the remainder of the command line (80 characters maximum) as its argument. It is the responsibility of the USRERUN routine to parse this command line (if needed).

The USRERUN subroutine looks like:

```
SUBROUTINE USRERUN (*, ITABLE, CLINE)
```

ITABLE is (as usual) the entire NOVA table structure. CLINE is the (80 character maximum) remainder of the command line entered with the ERUN command e.g. CLINE would be set to 'This is a test' if the command entered were

```
ERUN This is a test.
```

The subroutine should take the alternate return (RETURN 1) if there is an error (although currently this has no effect other than printing out a message).

See also the SRUN command.

$EVENTTIME

The time at which an event was generated (set by the MIDAS frontend code) is now available in the variable $EVENTTIME. It is an INTEGER*4 value (in msec -

relative to some arbitrary zero). If you want to make use of this value, you must define the variable $EVENNTTIME in your tables, although the value is filled in automatically when the event is retrieved.[1]

$EVTIMEBEG

The variable $EVTIMEBEG is the value of $EVENTTIME (see above) for the first event of a run (i.e. The Beginning-of-Run Event). The difference

```
$EVENTTIME-$EVTIMEBEG
```

is (of course) the relative time of an event during a run. Some experiments require finer gradation than this - if you need this then you must define a user function to reset the value of $EVTIMEBEG. For example, if Event Type 4 specifies the beginning of a new "burst", you could define the following.

```
RESET_TIME=USR01(RESET_TIME, $TABLE, $EVENTTIME)
```

Your OPSEQ would include

```
IF ($EVENTTYPE == 4) THEN
        EVAL RESET_TIME
ENDIF
```

and your USR01 function would look like

```
SUBROUTINE USR01 (X, ITABLE, IEVTIME)
IPTR = NVVALPTR (ITABLE, '$EVTIMEBEG)
IF (IPTR .GT. 0) THEN
        ITABLE (IPTR) = IEVTIME
ENDIF
```

Event Type

In MIDAS and YBOS modes, the (MIDAS) frontend code identifies events by two (16-bit) numbers - Event ID and Trigger mask. Typically, the Event ID is used to group together *classes* of events, and the Trigger Mask is used to distinguish among events within a class (e.g. Periodic/Scaler events are often assigned Event ID = 1, Physics Events are Event ID = 2).

In NOVA, the Event ID is used ONLY to select May/Must process modes (see $MAYMASK/$MUSTMASK) for groups of events. Within the OPSEQ, the NOVA Event TYPE (variable $EVENTTYPE) is taken from the Trigger Mask - the Event ID is

1 In version 2.2, $EVENTTIME and $EVTIMEBEG must be explicitly declared by the user. In version 2.3 these two variables are supplied automatically.

completely ignored. This means that for data to be analyzable by NOVA, the Trigger Mask must be unique (i.e. All events must have a different Trigger mask, regardless of Event ID).

By convention, the following Trigger masks are used for special "system" Events.

0x8000	Beginning of Run
0x8001	End of Run

Filenames

Originally, NOVA converted all filenames to lower case on Unix systems (since that is the convention for most people), and upper case for VMS (the reason for this is that NOVA itself is case insensitive, and so everything got converted to upper case, and then filenames on Unix systems got converted back to lower case). This turned out to be (a lot) more trouble than it was worth, and so the NOVA command parser was re-written to convert only those things which are important, and (in particular) to leave filenames alone.

Functions

The following new functions have been added to the repertoire which is available for use in expressions in NOVA. Details are found elsewhere in this document.

BNK2 (ASCII)	$RAW Bank pointer
BNK4 (ASCII)	$RAW4 Bank Pointer
I16TO32 (I1, I2)	32-bit from 16 bit
MAX(V1, V2)	Maximum
MIN (V1, V2)	Minimum
RAND (V)	Random number

gzipped files

It is possible to directly analyze gzipped (or any other compressed files) in NOVA (Unix systems only). It is not necessary to gunzip them first. To do this

1.Create a named pipe (fifo) using the Unix command

```
mkfifo /tmp/pewg
```

These "files" can exist anywhere, although it is traditional to put them in the /tmp directory (it makes it easier to keep track of them). Also, the name that you use should be unique (e.g. You could use your username to remind yourself who the file belongs to).

2.In NOVA, set your input source to be this fifo.

```
IMODE MIDAS          (or whatever is appropriate).
OPEN /FILE=/tmp/pewg
EA
```

3. In another window, run the appropriate 'unzip' program and direct the output to this named pipe.

```
gunzip < zipped_file > /tmp/pewg
```

I16TO32

The I16TO32 function is available to treat two 16-bit words as a 32 bit quantity.

```
I4VAL=I16TO32($RAW(1),$RAW(2))
```

will set I4VAL to 65536*$RAW(1) + $RAW(2)

For the $RAW vector, this can sometimes be done by accessing the (equivalent) $RAW4 vector. The I16TO32 function can be used for any variables, however (not just $RAW). Additionally, accessing $RAW4 doesn't work if the alignment is incorrect (e.g. $RAW4(0) treats $RAW(0) and $RAW(1) as a 32 bit integer, but there is no element in $RAW4 which does the same for $RAW(1) and $RAW(2)).

IMODE

The following data formats are now supported, and are described elsewhere in this document.

```
IMODE KVI
IMODE OFDAQ
```

KVI

NOVA can now analyze data collected at KVI in Groningen. Use

```
IMODE KVI
```

to select this input format.

LBANK {n, {m}}

The LBANK command will list the data banks in an event. If 'n' is entered, the command will first wait for the first event of type 'n' to appear (it will wait for at most 'm' events, and then complain if no such events appear - the default is 50 events). If n is not given, the event currently being analyzed is used. Output consists of the bank name, type, offset and length. Both offset and length are in INTEGER*2 words (i.e. They refer to $RAW, not $RAW4) regardless of the bank type. Also, offset and length refer to the data portion of the event only.

LBANK List bank structure of the current event.
LBANK 3 List bank structure of the next event type 3. Wait for at most 50 events (default).
LBANK 3 100 List bank structure of the next event type 3. Wait for at most 100 events to find one.

LOAD

The LOAD (and also DUMP) commands have been modified to use the environment variable NOVADUMPDIR. See the discussion of this variable for details.

MAX/MIN

MIN and MAX functions have been defined (2 arguments only). The syntax is

V=MAX(V1. V2)

Arguments may be either Real or Integer (both must be the same), and the return value is the same type as the arguments.

If you need more than 2 arguments, you have to combine them like

V=MAX(V1,MAX(V2,MAX(V3,V4)))

MIDAS

IMODE MIDAS is now accepted as defining the input format as that produced by the (TRIUMF-standard) MIDAS data acquisition system (previously, IMODE NET was used - it is still accepted). If there is anyone who really needs to use the old format (which was previously used only in a few TRIUMF experiments like E121 and E208), you can use IMODE OLDMIDAS.

MIDAS/YBOS

A number of comments are common to both MIDAS and YBOS event formats (they are both produced by a MIDAS front end - the only difference is in the Bank format of the data produced).

1. Several different "experiments" can be defined (in the file /etc/exptab). Additionally, an analyzer (NOVA) can connect to an experiment either on the local machine or on a

remote machine. This is specified in the OPEN command to NOVA.

   1. OPEN If no parameters are given, the local machine must contain only a single entry in exptab, and this one will be selected. If there is more than one, you will get a message listing the options, and asking you to select one, but you won't be able to do so! This will also 'break' the connection with the analysis sub-process. You will have to 'novakill' and 'novastart' again.
   2. OPEN local This is equivalent to the above, and indicates that the event source is a shared memory segment on the local machine (i.e. The same one as you are running NOVA on).
   3. OPEN local:expt This syntax selects the experiment called 'expt' (as specified in exptab) on the local machine. You MUST use this syntax if there is more than one experiment defined in exptab.
   4. OPEN machine:expt This syntax selects the experiment called 'expt' running on a remote computer called 'machine'.
   5. OPEN filename If the text after the OPEN command does not contain a colon (and is not the word 'local') it assumed to be a diskfile, not a realtime data acquisition process.

2. MIDAS (and YBOS) event formats do not contain a Run Number. The run number appears ONLY in the Beginning-of-Run (BOR) event, which contains also a dump of the ODB (online database). Thus, if you want access to the run number, you should make sure that you specify (in the /Logger/Channel/n/Settings directory in the ODB), that you want a dump of the ODB. When running online, NOVA periodically requests the run number from the ODB, so everything works as expected.
3. Two variables ($EVENTTIME and $EVTIMEBEG) are available to deal with the time of an event (filled in by the MIDAS frontend code). See the description of $EVENTTIME above.
4. Both MIDAS and YBOS event formats make use of the Event ID and Trigger Mask in specific ways in NOVA. See the discussion of Event Type above.
5. The LBANK command and the BNK2/BNK4 functions are valid only for MIDAS/YBOS Event structures.
6. MIDAS buffers events internally, and gives them to analysis tasks on request. Thus, it can occur that (for example, after a hardware change), the events returned to NOVA are "old" (corresponding to the situation before the change took place). The EA command has been modified to flush events before requesting them. Thus, this situation (retrieving old events) should not occur. If you suspect that it is, please let me know.

Multiple Commands

You can now enter multiple commands (separated by semi-colons) on a single command line (maximum 255 characters). These commands are executed sequentially (unless an error occurs, in which case execution is terminated prematurely).

This can be useful in conjunction with the command recall function, when you want to execute several commands repeatedly (up-arrow recalls the entire line, which can then be re-executed). Additionally, the REPT command can specify a succession of commands to be executed repeatedly.

Names

Names in NOVA may now be up to 16 characters long (the previous limit was 12). This makes it easier to choose your variable name in such a way that it reminds you what the variable is. Note that the DESCRIBE command is also available to further document the usage of a variable.

NOVABATCHSTATUS

See the discussion under NOVAWAIT.

NOVADATADIR

The environment variable NOVADATADIR specifies the directory/path where raw data files are stored (i.e. Files that you specify with the OPEN command). If it is not set, the current default directory is used.

Any filename which contains a directory specifier ('/' for Unix, '[' for VMS) will be used 'as is'. If the name does not contain a directory specifier, the string specified by NOVADATADIR will be tacked on the front of a filename before the system tries to open it.

NOVADUMPDIR

Both the DUMP and LOAD commands have been modified to use the environment variable (VMS logical name) NOVADUMPDIR as the directory/path to find dump files. The default (if this variable is not set) is to use the current directory. This makes it possible to keep dump files in a specific directory without having to worry about which directory is current when you run NOVA.

Before you enter NOVA (with the 'nova' command) you should

```
setenv NOVADUMPDIR pathname.
```

NOVAMAIN

The environment variable NOVAMAIN is used by the NOVASTART command to specify the location of the user program novamain.exe to run. It allows you to be sure that you run the correct version of the program regardless of your default directory. It should be defined as the full path of the file to be run, such as

```
setenv NOVAMAIN /usr/users/pewg/test/novamain.exe
```

The program to run is selected as follows:

1 NOVAMAIN environment variable (if defined).
2 novamain.exe in the current default directory (if NOVAMAIN is not set).
3 novamain.exe from NOVADIR.

NOVAPRINT

The NOVAPRINT environment variable is used by the HARDCOPY command to print plots on a hardcopy device. It is a character string. If the characters %f appear in the string, they will be replaced by the (internally generated) filename before being executed by the operating system.

This allows you to over-ride the default print command. For example, many people like to use the 'hprint' command to print things to a printer. You would do this by defining

```
setenv NOVAPRINT "hprint -w %f"
```

If the string %f does <u>not</u> appear in the NOVAPRINT command, the filename will be added to the end of the command. Note that double quotes are required if the replacement consists of more than one word.

NOVAWAIT

The NOVAWAIT command is typically used in batch jobs to wait for analysis of a run to complete before carrying on (e.g. DUMP results to a file and begin the next run). If an error occurs (e.g. You specify the wrong input file), things never get started properly, and hence the end-of-run condition never occurs.

NOVAWAIT now waits for 5 minutes, and monitors the number of events analyzed. If this number never changes, it is assumed that something is "broken", and NOVAWAIT exits anyway, first setting the environment variable NOVABATCHSTATUS to STALLED. Your shell script could test the value of this variable and take appropriate action (it will be set to OK if things are working).

OFDAQ

Some experiments (to my knowledge, only E497, although there might be others) collected data in the past using the DAQ format. To analyze tapes from these experiments, you must specify

```
IMODE OFDAQ
```

RAND (x)

The RAND function returns a (pseudo-)random number between 0 and 1. The argument x is ignored (although must be present).

$RAW
$RAW4
$RAWFP

The raw data in NOVA can now be accessed in several ways.

$RAW treats the raw data as INTEGER*2 words.
$RAW4 treats the raw data as INTEGER*4 longwords.
$RAWFP treats the raw data as REAL*4 Floating Point values.

Note that the actual data is the same in all cases (the current event). The difference is only in how NOVA interprets it.

REPT {n} command

The REPT command indefinitely repeats the remainder of the command line (including multiple commands if present) every n msec (default = 1000). One use of this command is to implement a 'live' spectrum display, where the display updates in real time. The command

```
REPT 500 DSP specname
```

will display the spectrum 'specname', wait 500 msec and then do it again.

The REPT command is terminated by hitting any key on the keyboarcd (just like the STAT command).

$SCLROFFSET

If the variable $SCLROFFSET is defined, it indicates the position in the $RAW event of the first scaler value. This is included to deal with events where there are headers (like the EVID bank discussed under YBOS), and hence scalers do not start at the first word of an event (the default USR00 event assumes that this is the case). If not defined, $SCLROFFSET defaults to 0.

SET

AXISMODE=n

For XWINDOWS display (not DECWINDOWS), the mapping of the data onto the axes can be specified as one of 3 ways.

0 The lower and upper limits of the axis will be "even" numbers (i.e. Integral multiples of the distance between Tic marks). The axis will appear as an integral number of tic marks. This is the default.

1 The lower limit of the axis will be an "even number" (i.e. Will correspond to a tic mark). The upper axis limit will be the largest data value, and not necessarily such an "even number".

2 The lower axis limit will be the smallest data value and the upper limit will be the largest data value. Thus, neither end of the axis will necessarily correspond to a tic mark, but the data range will completely fill the displayed axis.

Note that the tic marks are always chosen to represent a "nice" data value.

FILECOMP=n

This sets the level of compression used in generating Dump files. Possible values are 0 (no compression), 1 (the Default) and 2. I have found (based on limited experience) that Level 1 is adequate - the gain from using level 2 is minimal, and results in Loads and Dumps being quite a bit slower.

Note, though, that even uncompressed dump files (FILECOMP=0) are not compatible with earlier versions of NOVA. There is no way to load a "new format" dump file (even uncompressed) into an older version.

HISTORY=n

This flag controls which commands are entered on the NOVA command history list (those of you who have used NOVA for a while know that this has been the source of much confusion/controversy). Note that any command which does *not* come from the command history list, or *is* recalled from the history list and edited, is automatically entered as the most recent command on the list. This flag controls only those commands which are recalled are executed unchanged.

0 Recalled commands are *never* added to the history list. You can "fool" the system into thinking that a command has been edited by typing a single space before executing a recalled command.

1 Recalled commands are *always* added to the history list. This mode has the disadvantage that repeatedly recalling commands can fill up the history list with many copies of the same commands.

2 Recalled commands are added to the history list if they are NOT the most recent command on the list (i.e. If you recall the most recent command with a single up

arrow, it will *not* be added to the list, but if you recall an earlier command by typing up arrow more than once, it *will* be). This mode has the (distinct) advantage that a single up arrow always gives you the command you executed most recently. This is the default.

Spectrum Data

The /XDATA and /YDATA variables specified when a spectrum is defined can now be specified as Variable GROUPS. If you do this, a spectrum will be incremented *more than once* when the INCR statement is executed.

One use for this (and in fact the reason it was included), is to merge data from several detectors (presumably gain-matched) into a single spectrum.

For a 1D spectrum, the spectrum is incremented once for each member in the variable group specified for /XDATA.

For a 2D spectrum, either the X or Y data variable (or both) may be specified as a group. If only one of them is, the INCR statement loops over that one, using the single value for the other. If both are groups, the sizes MUST be the same, and corresponding elements of each are used.

SREAD

The SREAD command allows you to read data into a NOVA spectrum. The qualifiers are identical to those of SWRITE.

SRUN

The SRUN command is identical to the ERUN command, except that the subroutine which is called is USRSRUN rather than USRERUN. See the discussion of ERUN above for more details.

$STATPAGE

The variable group $STATPAGE contains a list of variables which will be listed by the STAT command (in addition to the usual system information about number of events processed etc.). Up to 51 variables may be included in this group. If a simple variable, only the value will be displayed. If a condition, the two software scalers (number of times tested and number of times passed) will be listed.

SWRITE

Several new options have been added to the SWRITE command.

```
SWRITE /FILE=path spectrum
```

This allows the filename to contain special characters (like /).

SWRITE/FULL

Underflow and overflow channels are output. This is the default for "normal" format output - this qualifier is useful mostly for PLOTDATA and MATLAB format output (where the default is not to output under/overflow channels).

SWRITE/MATLAB

The data is written in a format which can be easily read into MATLAB.

SWRITE/NOHEADER

The header lines (containing spectrum name, Axis limits etc.) are not output - just the data.

SWRITE/PLOTDATA

The data is output in a format which is easily read by PLOTDATA/PHYSICA.

$TABLE

The "variable" $TABLE contains the entire table structure of NOVA, and is useful in USR functions (in conjunction with functions like NVVALPTR described above). I mention it explicitly here only because (some)previous versions of the documentation erroneously referred to $TABLES - the variable is actually $TABLE (without the trailing S).

Tcl

A rudimentary (but still useful) Tcl interface to NOVA has been developed. Most of the control/monitoring functions (including start/stop of a run, spectrum display etc.) are available using the mouse instead of the command line. Definition of new variables (or editing existing definitions) is NOT available. Note that the command line interface can still be used (even at the same time - from a different window) to add new definitions.

The Tcl interface is accessed by running the program[2]

1987a

Useful Functions

2 For extra credit, you are invited to speculate on why this is an appropriate name for this program.

The following functions might be useful in USR routines to get at particular information. If you want to use these functions, your USR routine must contain the argument $TABLE, which gives you access to the complete table structure of NOVA.

I = NVRDCOND (ITABLE, CNAME, IVALUE, NUMTOT, NUMTRUE)

This function returns the current value, number of times tested and number of times TRUE for the condition named CNAME. It might be used in a URSDA or USRERUN routine to calculate statistics at the end of a run. The return value (I) is negative if the condition cannot be found.

I = NVRDIF (ITABLE, IWHICH, ILINE, NUMTOT, NUMTRUE)

This is similar to the previous function, and returns in addition the OPSEQ line number for the IWHCH'th IF statement in the OPSEQ (the first one is IWHICH=1).

IPTR = NVVALPTR (ITABLE, CNAME)

This function returns a pointer (Index into ITABLE) to the location where the *value* of the variable CNAME is stored (i.e. The value of the variable CNAME is stored in ITABLE (IPTR)).

User Functions

You are now allowed up to 100 User Functions (named USR00 to USR99). Each may have up to 25 arguments.

For compatibility with existing command files, the names USR0 --> USR9 are still accepted as equivalent to USR00 --> USR09, and USRA --> USRZ are equivalent to USR10 --> USR35 (in NOVA command files). The actual FORTRAN subroutines, though, must be edited and recompiled using the new names.

YBOS

This is not really something new, but something not previously documented. The YBOS event format does not include an event header containing lots of useful information (like Event Type/Trigger Mask / Time). For this reason, all YBOS events should include a bank called EVID (it MUST be the FIRST bank in the event), which contains a copy of the MIDAS event header. If it is not present, the Event type will always be 0, the event length will be set correctly, but the Sequence Number and event time will not be accessible.

$ZALLVARS

The variable group $ZALLVARS contains the names of any variables which should be cleared to zero by the ZALL command. By default, spectra and condition counts are cleared by ZALL - variables are not.

Typically, the scalers-related variables OLDS, TOTS would be included in this group. It is defined in the usual way

```
ADDGROUP/TYPE=VAR $ZALLVARS OLDS TOTS ...
```