

NAME

changes – highlights of modifications for spec release 2.14

DESCRIPTION

These notes summarize many of the modifications made for spec release 2.14, as of January 8, 1991.

INCOMPATIBILITIES

Release 2.14 includes a number of changes that make certain prior usages obsolete. You must modify any home-grown macros to take into account the following changes:

- 1) The customized scan output hooks involving the macros `Plabel`, `Pout`, `Flabel` and `Fout` have had the position of the space character changed from before each string in the label to after each string. You must edit your macros if you assign values to any of the above or if you have written new scans based on the standard scan macros.
- 2) The argument of the `wait()` function, as programmed, did not match the documentation. The documentation stated that bit 1 represented motor status and bit 2 represented counter status. As programmed, these bit assignments were reversed. The program has been changed to match the documentation. You must edit your macros if you call the function `wait()` directly.
- 3) The `config` file now contains information describing each scaler's label, mnemonic, function and hardware type. The scaler-name array `S_NA[]` no longer exists. New string functions, `cnt_name(i)` and `cnt_mne(i)`, return the name and mnemonic of scaler `i`. You must edit your macros if you make any reference to the symbol `S_NA[]`.
- 4) The value returned by the `getcounts` command in `S[sec]` is now in seconds, rather than milliseconds. You must edit your macros if you divide `S[0]` by 1000 anywhere.
- 5) The `yesno` macro has been replaced by a `yesno()` function. You must edit your macros if you invoke `yesno` anywhere.

USER INTERFACE

The `history` command now takes a numerical argument that limits the number of the previous commands displayed. In addition, if the the argument is negative, the recalled commands are displayed in reverse order. The number of input lines saved for history recall has been increased from 50 to 100.

High resolution scan plots are now available with VGA, EGA and Hercules graphics adapters on System V/386 and VENIX 286 platforms; with the X Window System Version 11 on PS/2 AIX, Ultrix and SunOS; and with SunView on SunOS. You should set a `GTERM` environment variable to `vga`, `ega`, `cga`, `herc`, `x11` or `sun` before invoking `spec` to select the appropriate graphics parameters. The `setplot` macro lets you select whether or not to use high resolution plotting and if so, how to depict the data (lines, points, dots).

Text-mode plotting has been moved from macros to built-in C code. The plots are drawn much faster and are now available with any terminal described by the system terminal database. The names of the macros used to do the screen plots and the real time plots (`splot` and `rplot`) have not changed, although terminal dependent macros such as `ansi_rplot` and `vt52_plot` have been eliminated.

The built-in symbols `ROWS` and `COLS` contain the screen size used for drawing the plots (and displaying help files). These are initially taken from the environment variables `LINES` and `COLUMNS`, or if unset, from the system terminal database entry described by your `TERM` environment variable. You may reassign values to the variables if you change your screen size (for example on a workstation using windows).

`spec` now prevents multiple instances of the program being run using the same state files.

When `spec` is run from pseudo ttys (for example, on an X Windows workstation or over the network), the state file is associated with the name `ttyp#`, rather than the actual name of the pseudo tty.

You can now create a state not associated with any tty, by starting `spec` with the `-T name` command line option.

A new variable, `CWD`, keeps track of the current working directory of the running program. The `pwd` macro has been changed accordingly to simply print the value of `CWD`, rather than spawn a subshell to execute the command.

The help utility has been entirely rewritten. Help files are formatted on the fly to fit the size of the screen or workstation window. In addition, standout mode is used to highlight text that would be printed in an alternate font.

The number of nested command files allowed has been increased from five to eight.

MACROS

If a macro named `begin_mac` is defined, it will be run after the startup command files are read but before input is read from the keyboard.

The motor scans `ascan`, `a2scan` and `a3scan` have been rewritten in terms of a general macro called `_ascan` that scans an arbitrary number of motors. An `a4scan` that calls `_ascan` is now also provided, as is a `d4scan`. These scans introduce a number of new global variables (`_nm`, `_m[]`, `_f[]`, `_s[]`, `_d[]`) that you should avoid using in your own macros.

A new standard macro called `chk_beam` has been inserted into the macros that do normal, updated and powder-mode counting during scans. In general, these macros are now written along the lines of:

```
for (;;) {
    count_func(count_time)
    waitcount
    getcounts
    chk_beam
}
```

where `count_func()` is `tcount()`, `mcount()`, etc. By default, `chk_beam` is defined as `break`, meaning the infinite loop is immediately exited after counting. However, you can define `chk_beam` to not break out of the loop unless the monitor counts, for example, are above some threshold. Such a feature can be useful at a synchrotron where beam dumps or fills that occur during unattended data taking could result in many useless scans. See the `count.mac` macro source file for an example of a complete `chk_beam` macro.

A new option to the `setscans` macro asks you to enter a value for a sleep (or settle) time to delay before counting at each point in a scan. The value is stored in the global variable `_sleep` (which you can also change simply by assigning it a value). The scans all execute a `sleep(_sleep)` call before counting. Set `_sleep` to zero for no delay.

The macro that continues aborted scans, `scan_on`, now accepts an argument to adjust the current point number in the scan (except for mesh scans). For example, `scan_on -1` redoes the previous point, while `scan_on 3` skips the next three points. Giving a very large number will force the end-of-scan plots, etc., without taking any more data points.

A new set of macros lets you define a set of scans to be done together. The scans must all be of the same type and must be in terms of a single scan variable. You may however define regions, point densities and count times differently for each scan. The `setreg` macro (for *set regions*) asks you to configure each scan. The `doreg` macro executes the group of scans. A number of global variables beginning with the characters `_reg_` have been introduced to keep

track of the region scan configuration.

The `ct` macro, if aborted with a `^C`, will now execute a `cleanup` macro that reads the scalers and shows the counts that were accumulated before the `^C`.

The names `det` and `mon` are no longer built into the macros, since scaler mnemonics may now be set to arbitrary values in the `config` file.

The `show_cnts` and `uct` macros have been rewritten to show the accumulated counts in all the configured scalers.

The `_head` and `_loop` macros have been modified to include counts from all configured scaler channels in the data file. Previously, only the detector and either monitor or seconds scaler contents were saved for each point.

The `MON_RATE` global variable has been eliminated from the standard macros due to lack of interest.

Many of the standard macros involving motor positions have had hooks inserted to support a feature involving customizable position *units*. This feature is not currently available to most users of `spec`, and the changes will not affect the functionality of any of the macros. However, the names `_units`, `_setfmts`, `_suffix`, `P`, `U`, `UL`, `UNITS`, `PFMT` and `VFMT` are used by this feature and are henceforth reserved for use by the standard macros.

FUNCTIONS

The function `motor_par()` can now return several parameters from the `config` file. Also, the motor velocity and the number of steps to use for backlash can be set from user level. See the *motors* help file for details.

A new function, `ca_fna()`, allows arbitrary CAMAC access. See the *funcs* help file for details. At present, it is up to the user to avoid addressing slots that are being used for motor control or counting.

A new function `gpib_poll()`, returns the serial poll status of a device on the GPIB bus. This function is currently only available with National Instruments GPIB controllers. See the *gpib* help file for details.

New functions, `cnt_name(i)` and `cnt_mne(i)`, return the name and mnemonic of scalers as set in the `config` file.

A new function, `plot_cntl(mode)`, is used to control the built-in plotting features. The argument `mode` is a string of comma- or space-delimited options. See the *funcs* help file for details.

A new function `tty_cntl(mode)` is used to send terminal-specific special sequences to the screen. The argument `mode` is a string. Recognized strings are "ce" (clear to end of line), "cl" (clear screen and home cursor), "cd" (clear to end of display), "so" (begin standout mode), "se" (end standout mode) and "ho" (home cursor).

A new function `tty_move(x, y [, s])` positions the cursor at column `x` and row `y` of the screen, and prints the optional string `s` at that position. The upper-left corner of the screen is column 0 and row 0. If `x` or `y` are negative, the position is taken from the left or bottom of the screen. Output written by this function goes only to the `tty` device, regardless of what devices have been turned on with the `on()` function.

A new function `plot_move(x, y [, s])` works just as `tty_move()`, above, when high-resolution graphics mode is off. When high-resolution graphics mode is on, `plot_move()` draws to the graphics screen.

A new function `plot_range(xmin, xmax, ymin, ymax)` sets the ranges of the internally generated plots. If any of the arguments is the string "auto" the value is determined by the

appropriate value from the current points.

A new function `plot_pts(xcol, start, npts)` plots the current data from the built-in plot array either on the screen if graphics mode is off, or on the high-resolution graphics screen or window if graphics mode is on. The plot starts with point number `start` and uses `npts` points. The `xcol` parameter will be used in the future, when the number of values stored for each point is increased, to designate which value to use for `x`.

A new function `yesno([s,] x)` prompts the user with the optional string `s`, for a yes or no response. The function returns 1 if the user answers with a string beginning with `Y, y` or `1`. The function returns `-1` if the user answers with a string beginning with something else. The value of `x` is returned if the user simply enters a return. Usage might be,

```
flag = yesno("Show updated moving", flag)
```

which produces (if `flag` is nonzero),

```
Show updated moving (YES)?
```

GEOMETRY CODE

The geometry code and macros for the various *surf* diffractometer configurations, as used at NSLS X20 and X22, CEA in France and at Exxon have been rearranged to ease maintenance and installation of each version.

The NSLS X20 monochromator macros and geometry code have been isolated from the four-circle code, so that references to the monochromator will only be made if the monochromator option is selected at installation time.

The four-circle code now allows sector 1 to be used in modes 4, 5 and 6.

The four-circle code now prints an error message when a user tries to calculate reciprocal lattice positions with both *chi* and *phi* at zero in **zone** mode, as the current method of doing the calculations cannot handle that case.

The three-circle mode with *phi*=0 has been generalized to a *phi*-fixed mode. Also, the frozen modes now include *phi* fixed.

A new scan macro, `aziscan`, is available in four-circle mode to scan the azimuthal angle in **azimuthal** mode.

Bugs in the *zaxis* geometry code calculations have been fixed.

HARDWARE

The internal code implementing hardware support in `spec` has been substantially revised to simplify adding support for new types of hardware. The revisions for the most part should be transparent to the user. If a particular hardware device fails to perform as it did in previous versions, notify CSS at once, (unless the difference is an improvement).

Support for the Kinetic Systems 3912 CAMAC crate controller is now provided on DEC Q-Bus based systems.

Support for the Kinetic Systems 3988 GPIB CAMAC crate controller is now included. This module can only be used if a National Instruments GPIB controller is also installed. The board can be used in either interrupt-driven or polled mode. Interrupt driven mode is preferred, but if interrupts are lost in the GPIB driver, polled mode can be selected.

Support for the Ortec 974/994/995 counters and counter/timers is now provided over both GPIB and RS-232C interfaces.

Support for the Klinger MC4 motor controller is now provided over both GPIB and RS-232C interfaces.

Support for the Advanced Control Systems MCU-2 motor controller is now provided over an RS-232C interface.

Persistent timing problems with the Kinetic Systems 3388 GPIB CAMAC module seem to be fixed now.

spec now recognizes the Joerger SMC 2601 motor controller, which allows for programmable velocity and acceleration.

The code for the TS201 CAMAC timer/counter model now uses timer 2 to gate counter 2, making it no longer necessary to use an external cable to gate counter 2.

The “new and improved” E500 modules that use the F(10) A(0) command to clear the module LAM are now supported.

The E500 LAM service routine has been modified to use fewer FNA commands and to recheck for additional LAMs from the E500 before returning. The latter change improves performance when several motors in the same E500 complete their motions at nearly the same time.

The code that reads the LeCroy 2301 MCA unit has been changed to read the device faster. (On a 25 MHz 386 using the DSP 6001, the change is from 0.36 seconds previously to 0.16 seconds now for reading 1024 channels.)

Many options for use of the National Instruments GPIB controllers formerly set by each site using the *ibconf* program are now set by spec at run time. Specifically, spec sets the timeout period to 3 seconds, the EOS byte to a newline (‘\n’), reads to be terminated on EOS using an eight-bit compare, REN to be asserted and the board to be the system controller. The board’s primary address is set to 0 and the signal to be sent on service request (if using the board with a GPIB-to-CAMAC controller) is set appropriately. You may still use *ibconf* to set whether EOI will be asserted with EOS and/or with the last byte of a write.

The length of a string that can be read using `gplib_get()` has been increased from 64 to 255 bytes.

The GPIB functions `gplib_get()` and `gplib_put()` no longer generate an *interface clear* (IFC) message on the bus, unless a previous transaction had resulted in an error. The old code that sent an IFC before each transaction not only slowed down I/O, but also caused some devices to time out. In addition, a brief delay is inserted after an IFC is sent to accommodate slow devices. (These devices did not follow GPIB standards and used the IFC message to reset internal functions rather than just interface functions. The device were often not yet ready to respond to the next interface message after the IFC period was over.)

If the National Instruments GPIB code has errors on two transactions in a row, the GPIB device is reset by closing and reopening the `/dev/gplib0` (or whatever is configured) special file. If this feature causes more problems than it solves, inform CSS immediately so we may begin work on alternatives.

Use of the `ser_get()` function to read RS-232C devices has been changed to increase its flexibility. See the *serial* help file for details.

spec used to silently enforce a rule that only one GPIB controller, MCA or timer was allowed at a time. A message is now printed if the *config* file requests more than one GPIB controller, CAMAC controller, timer or MCA. The first device of each type encountered in the *config* file is the one that will be used.

Now that scaler numbers are assigned in the *config* file, it is possible to configure more than one type of scaler device at a time. Each scaler should use hardware gating from the master timer. Software gating does take place, although there can be substantial latency between receipt of the time-complete signal from the master timer and disabling of the scalars. Note

also that for a number of combined timer/counter units, the scaler assignment for the seconds and monitor channel are hard coded into `spec`.

For versions of `spec` on most platforms, memory is allocated for the maximum number of motors (and scalers) at program startup, so that it is no longer necessary to restart the program to increase the number of motors (or scalers) after editing the `config` file.

UTILITIES

A stand-alone version of the C-PLOT `scans.4` user function, called `scans`, is now provided for extracting a data set from a `spec` data file. See the `README` file in the `aux` subdirectory of the distribution.

ADMINISTRATION

Version 2.14 has a new installation procedure, that is based on the new `Install` script in the distribution directory. A `README` file explains the use of the script.

The CAMAC drivers on all systems should be replaced with the new versions in the `drivers` subdirectory of the distribution. Versions of `spec` prior to version 2.14 should work with the new drivers, but `spec` 2.14 will not work with old versions of the drivers. Changes to the drivers are as follows: The new drivers can be configured at run time for the signal to send to the user process when a LAM is generated. The Q and X response of each dataway transaction is now communicated by the driver to the user process in a way that does not depend on which CAMAC controller is installed.

The `camac.h` include file has been revised to accommodate CAMAC drivers on BSD 4.3 and ULTRIX operating systems and to include added functionality. The copy of `camac.h` in the directory `/usr/include` will be updated automatically if you install the new CAMAC driver.

On System V/386 systems, a new `nap` driver is included that performs fractional second sleeps. Run the shell script `install_nap` in the `drivers` subdirectory as super user to perform the installation.

The GPIB driver for the National Instruments PCII board for the VENIX 286 system must be replaced with the new version of the driver in this release. Run the script `install_gpib` in the `drivers` subdirectory as super user to perform the installation.

Another command file can be automatically read on startup. This file is named `site.mac` and resides in the auxiliary file directory (SPEC) as, for example, `/usr/lib/spec.d/site.mac`. If this file exists, it will be read every time a user starts the program, not just when starting fresh. The `site.mac` file is also read when the `newmac` macro is run.

More work has been done on the `edconf` program. In particular, if you exit before saving any changes, you will be asked to confirm whether you really want to quit. Also, there is now a fourth screen used to configure scaler assignments. The number of configurable scalers on the devices screen has been increased to three. The CAMAC slot assignments in the `config` file now begin with `CA_` rather than `C_`, as in the old versions. The old format is still recognized on reading, but the new form will be written out. Arrow key sequences are now read from the terminal capability data base, and so `edconf` can be used with arrow keys on non-vt100-compatible terminals. In addition the `vi` hjkl motion keys are now recognized.

The hardware support in `spec` has been completely modularized. The `u_hdw.c` file contains a simplified hook mechanism to select the installed hardware. All other references to a particular hardware device are now contained in a single source file, allowing sites to add support for new devices, without requiring access to the complete source code. (At present the `edconf` program cannot be changed at site to add new hardware types.)

When `spec` starts out it executes a call to `nice(-20)`. This call raises the priority of the process, but only when being run by root or if the process is owned by root and has set-user-id execute mode set. After raising the priority, `spec` resets the effective user and group ids to

that of the real user, so there is no danger of the user spawning subshells or creating files as root. If `spec` is not set-user-id root, it will behave as before.

If the *settings* file can't be opened for writing when `spec` starts up, or if there are read or write errors accessing that file, `spec` switches to simulate mode and suggests the user get assistance.

MISCELLANEOUS

`spec` is now available for the IBM PS/2 running the AIX 1.2 operating system. CAMAC support is currently only through a Kinetic Systems 3988 GPIB-to-CAMAC crate controller.

`spec` is now available for DEC MicroVax computers running ULTRIX or UCB operating systems. CAMAC drivers are available for either the Kinetic Systems 3912 or DSP DCC-11 crate controllers.

`spec` is now available for SCO Xenix 286 2.2.3 and SCO Xenix 2.3.2 operating systems. (Use on a 286 platform is discouraged, though.)

The motor number indirection array `mA[]` is now reinitialized when the `config` macro is invoked, using the `_assign` macro that is defined in each geometry's standard macro file. Thus, if motor numbers are reassigned while editing the *config* file, the macros that list motors in specific order (*wa*, *wh*, ...) will still list the motors in the same order after `_assign` is run.

The `print` command with no arguments will now print a newline. In previous versions, such usage was reported as a syntax error.

The number of command files that may be queued for reading has been increased from five to eight.

BUG FIXES

A design flaw, introduced in version 2.11, and having to do with reads of the motor controller registers, has been corrected. The problem was that `spec` could lose track of motor positions if the motor controller registers were changed using a remote control pod, or if they were cleared due to loss of power, while `spec` was running. If any of the commands that read the controller registers (`getangles`) or changed the *settings* file (`chg_dial()`, `chg_offset()`), were executed before executing a `sync` command, the motor positions would likely have gotten scrambled.

A bug, whereby the results calculated using the background subtraction feature were incorrect, has been fixed.

All motor mnemonics are now freed when executing the `reconfig` command and then reestablished on rereading of the *config* file. Previously, a motor mnemonic, not used in the geometry code, that was removed from the *config* file would remain as an immutable symbol until you ran `spec` with the `-f` flag for a fresh start.

A bug where motor control would hang when trying to start a motor on a Joerger SMC motor controller that had both limits open has been fixed.

A bug with the TS201 timer/counter where LAMs would sometimes be lost during updated counting has been fixed.

A bug with the TS201 timer/counter where readings of certain count times (30 seconds, for example) would indicate the wrong period (although the period over which the module counted was correct) has been fixed.

A bug where values assigned to variables declared `local` within a statement block sometimes affected variables of the same name outside the statement block has been fixed.

A potential bug where defining a macro recursively would cause a program crash has been fixed by limiting the amount of input push back to 64 Kbytes. (This limit is arbitrary and can be increased by CSS if found to be too low.)

A bug where *log* file names containing a slash were not treated as *log* files has been fixed.

An old bug, where reference to the string constant and then a number constant of the same value, particularly with a value of zero, would cause unexpected results, has been fixed. In previous versions, the code:

```
{
  a = "0"
  b = 0
  if (b) print "This should not happen"
}
```

would result in the message being printed.