

NAME

spec_par() – sets internal parameters

DESCRIPTION

The `spec_par()` function allows access to certain internal parameters of `spec` as described below. Most parameters can be set using the `-o` start-up option (see the `spec` help file). Some parameters can only be set using the start-up option. As of `spec` release 5.03.01, some hardware-related parameters can also be set in the `config` file, using the following syntax:

```
SPEC_PAR: epics_timeout = 2
```

The declarations have to be entered manually, but will be maintained by the `edconf` configuration editor. The declaration can be located anywhere in the file.

`spec_par("?")` – Displays all the parameters and their values. If the current value differs from the default value, the default value is displayed in parenthesis (as of release 5.08.02-8).

`spec_par("set_defaults")` – Sets all parameters to their default values.

`spec_par(par)` – Returns the current value of the parameter `par`.

`spec_par(par, value)` – Sets the internal parameter `par` to `value`.

The `spec_par()` parameters are:

"anka_remote_host", "anka_remote_node", "anka_locate_node" – These parameters are only visible when the ANKA RST Gamma hardware is configured. See the `anka` help file for an explanation of what they mean. These parameters can be set in the `config` file.

"auto_file_close" – The auto-file-close option is available to automatically close output files that haven't been accessed for some interval of time. The parameter units are hours, and the parameter can have nonintegral values. When the auto-close option is enabled, each time an `on()`, `off()`, `open()`, `close()` or `fprintf()` function is called, `spec` will check its list of opened output files. Any files which have not been written to for the length of time given by `value` hours will be closed. Enabling this option can help prevent errors when your macros or commands do not close files when appropriate, resulting in `spec` running out of resources to open additional files.

As files are opened automatically when sent output, auto-close mode operates transparently for the most part. However, if you change to a different working directory between the time the file is first opened and subsequently automatically closed, and if the file is not opened by an absolute path name, the next time you refer to the file, `spec` will reopen it using a relative path based on the current directory.

If `value` is zero, the mode is disabled. By default, the mode is initially disabled.

"auto_hdw_poll" – When automatic hardware polling is turned on, `spec` will automatically poll busy motor controllers, timers and acquisition devices to determine when they are finished. For some polled devices, `spec` needs to perform an action, such as starting a motor backlash move, when the device is finished with its current business. Without automatic hardware polling, a call to the `wait()` function is necessary. The default is for this mode to be on. A reason to turn it off may be to reduce the amount of debugging output during hardware debugging. Early versions of `spec` made more frequent use of interrupt-driven devices (certain motor controllers and timers), for which this mode is irrelevant.

"check_file_names" – The check-file-names option can prevent you from (accidentally) creating files with names containing nonstandard characters. When enabled, if a file name passed to the `on()`, `open()` or `fprintf()` functions contains any of the

characters `()[]{}|$\`*?;!&<>`", the space character, any control characters or any characters with the eighth bit set, and the file doesn't already exist, `spec` will print an error message and reset to command level. By default, this mode is on.

- "`confirm_quit`" – If set, `spec` prompts with a "Really quit?" message when the `quit` or `^D` commands are entered. The question must be answered in the affirmative to exit the program. The value for "`confirm_quit`" is not saved in the state file. The option must be set again on each `spec` invocation.
- "`elog_level`" – The level at which commands are saved to the `elog` error files is set using this option. At level 1, the default, only commands typed at the keyboard are logged. At level 2, the commands read from command files are also logged.
- "`elog_timestamp`" – The time interval for the optional time stamps for the `elog` error file capability is set using this option. The units of the "`elog_timestamp`" parameter are minutes. The default value is 5 minutes. Note, time stamps are only added before a command or error message is logged, so that the interval between time stamps can be greater than that specified if no commands are being typed or errors generated. However, for time-stamp intervals less than 1 second (or 0.1666 min), every command and error message will be preceded by a time stamp, and the time stamp will include fractional seconds in the epoch value.
- "`epics_timeout`" – Sets the default timeout for channel access `ca_pend_io()` calls on EPICS. The default value is 0.5 seconds. This option appears only when `spec` is linked with the EPICS channel access libraries. Timeout values for individual process variables can still be changed with the `epics_par()` function. This parameter can be set in the `config` file.
- "`fast_hdw_checks`" – When in server-mode, if set, the `spec` server will use an adaptive algorithm for polling hardware status. If the interval between client messages is less than one second, the server will use a faster polling frequency for checking for both hardware status and the next client message, thus providing a client with faster notifications of changes in hardware state. This parameter can be set in the `config` file. With its (documented) introduction in `spec` release 5.08.06-2, the mode is off by default.
- "`flush_interval`" – The flush-interval parameter controls how often `spec` flushes output to the hard disk or other output devices. Traditionally, `spec` flushed output after each print command. However, as some users report that this frequent flushing introduces considerable delays when the output device is to an NFS-mounted file system, the flushing interval can now be controlled. The "`flush_interval`" parameter specifies how many seconds to allow between output buffer flushing. The default value is 0.5 seconds. If the interval is set to zero, the traditional frequent-flushing behavior will be restored. Output to the screen is still flushed immediately. Output is also flushed each time the main `spec` prompt is issued.
- "`HKL_rounding`" – Traditionally, `spec` rounded values for H, K, and L (and other geometry values derived from motor positions) to five significant digits for configurations using reciprocal space calculations. As of release 4.03.01, the rounding is now turned off by default. It can be turned on using the command `spec_par("HKL_rounding", 1e5)` where the argument indicates the magnitude of the rounding, i.e., one part in 1e5, for example. Note, values with an absolute value less than 1e-10 are still rounded to zero whether or not the optional rounding is turned on.
- "`hdw_poll_interval`" – When the `wait()` function is called to wait for polled motors, timers or other acquisition devices to finish, `spec` sleeps for a small interval between each check of the hardware. Use this `spec_par()` option to change that interval. The

units of the parameter are milliseconds, and the default sleep time is 10msec. A value of zero is allowed, though not recommended if the computer is being used for anything else.

"keep_going" – Normally, when taking commands from a command file, **spec** resets to command level and the main interactive prompt when there are syntax errors in the file, certain floating point exceptions, references to unconfigured hardware, hardware access errors, along with a number of other errors. When the "keep_going" option is set, **spec** will keep reading and executing commands from a command file no matter what errors occur. When there is an error, the next line from the current command file will be read. Note, depending on where the error is in a file, reading subsequent lines may generate more errors, particularly if the error occurs inside a statement block.

"modify_step_size" – Normally, **spec** doesn't allow users to modify the motor step-size parameter with the `motor_par()` function, as the consequences are generally undesirable. However, in the rare circumstance that it is necessary, this parameter allows you to enable such modifications.

"old_shared" – With **spec** release 5.02.01, the structure of the shared array header was changed so that the data portion of the array would lie on a memory page boundary. To allow compatibility with applications built with the old header structure, the "old_shared" option can be set. However, this option can only be set as a `-o` command line start-up option, and the parameter is not saved in the state file. It must be set each time **spec** is invoked.

"parse_units" – When parsing of units is turned on, numbers typed as input to **spec**'s parser with one of the following suffixes appended will automatically be multiplied by the corresponding factor.

1r	= 57.2958	radian
1mr	= 0.0572958	milliradian
1d	= 1	degree
1md	= 0.001	millidegree
1mm	= 1	millimeter
1um	= 0.001	micrometer
1m	= 0.0166667	minute
1s	= 0.000277778	second

Note, however, suffixes on numbers converted from strings or entered using the `get_val()` function are not parsed. The only known use for unit-suffix parsing is with the user-contributed macros in the file `macros/units.mac`. These macros require that unit suffixes be supplied for all motor position arguments in the standard **spec** macros. The default is for this mode to be off.

"show_prdef_files" – When this mode is on, the source file for each macro definition is displayed with the `prdef` command. The default is for this mode to be on.

"specwiz" – Allows **spec** administrators to gain access to motors locked out in the `config` file to ordinary users. This feature allows the administrator to position the motors without having first to go into the configuration editor to change access modes. Entering `spec_par("specwiz", 1)` causes **spec** to prompt for the "Wizard's password". If entered correctly, the characters `_WIZ` are appended to the **spec** prompt to remind the administrator of the special powers, and motors configured with protected status can be moved. Entering `spec_par("specwiz", 0)` disables the special mode.

spec looks for the encrypted password belonging to the `spec_wiz` (or `specwiz`) user in the files `SPECD/passwd`, `/etc/shadow`, and `/etc/passwd` in turn. If a shadow

password file is used, ordinary users won't be able to read it, and the normal password file won't contain the encrypted password. The spec distribution includes a *wiz_passwd* utility (as of release 4.03.07) that can be used to create a *passwd* file in the spec auxiliary file directory that contains just the entry for the spec_wiz user. The *SPECD/passwd* file should probably be owned and writable only by root or the spec administrator.

Note, the standard macros *onwiz* and *offwiz* are convenient wrappers for the *spec_wiz* feature.

"*use_sem_undo*" - This flag relates to whether the *SEM_UNDO* flag is set when semaphores are used. It exists to get around a memory leak bug observed with some releases of the Solaris 2 operating system. The flag should be ignored unless you are instructed otherwise by CSS.

"*warn_not_at_pos*" - When enabled, spec prints a warning message whenever a motor doesn't reach its final position (as of release 5.08.02-8).