

**NAME**

counting – timer/scaler/mca commands and variables

**BUILT-IN COMMANDS AND FUNCTIONS**

`mcoun(t)` – Begins counting for *t* monitor counts. Returns **true**.

`tcoun(t)` – Begins counting for *t* seconds. Returns **true**.

`cnt_mne(i)` – Returns the string mnemonic of counter *i* as given in the configuration file.

`cnt_name(i)` – Returns the string name of counter *i* as given in the configuration file.

`cnt_num(mne)` – Returns the counter number corresponding to the counter mnemonic *mne*, or -1 if there is no such counter configured.

`counter_par(i, s [, v])` – Returns or sets configuration parameters for counter *i*. Recognized values for the string *s* include:

"monitor" – Reassigns the scaler channel used as the monitor-preset counter to the channel associated with counter *mne*. Rereading the hardware configuration file with the `config` macro or `reconfig` command will restore the monitor channel to that set in the *config* file. This feature is only available with certain counters. Counters that currently support this feature include the KS 3640 CAMAC counter, the Joerger VSC16/8 VME scaler and the generic EPICS scaler.

"timer" – Reassigns the scaler channel used as the time-preset counter to the channel associated with counter *mne*. Rereading the hardware configuration file with the `config` macro or `reconfig` command will restore the time channel to that set in the *config* file. Currently no timers support this feature.

"controller" – returns a string containing the controller name of the specified counter. The controller names are those used in `spec`'s *config* files.

"unit" – returns the unit number of the specified counter. Each counter controller unit may contain more than one counter channel.

"channel" – returns the channel number of the specified counter.

"responsive" – returns a nonzero value if the counter responded to an initial presence test or appears otherwise to be working.

"disable" – returns a nonzero value if the counter has been disabled by software. If *v* is given and is nonzero, then the counter is disabled. If *v* is given and is zero, the counter becomes no longer disabled. A disabled counter channel will not be accessed by any of `spec`'s commands. Any `cdef()`-defined macros will automatically exclude the portions of the macro keyed to the particular counter when the counter is software disabled.

`getcounts` – Loads the built-in array `S[]` with the scaler counts.

**VARIABLES**

`S[]` – Built-in array that holds scaler contents after a `getcounts` command.

`COUNT` – Global variable that holds the default count time used by the `ct`, `uct` and `uctn` macros.

`COUNT_TIME` – Global variable that holds the actual count time used by the `ct`, `uct` and `uctn` macros.

`DET` – Global variable that is set to the channel to be used as the detector during scans, usually set to `det`.

`MON` – Global variable that is set to channel to be treated as the monitor during scans, usually set to `mon`.

MON\_RATE – Global variable that holds the value  $S[\text{MON}]/S[\text{sec}]$  and is updated in the `count` macro.

Scaler channel assignments and scaler mnemonics are set in the `config` file. The standard assignments for the first three channels are:

```
sec = channel 0
mon = channel 1
det = channel 2
```

## MACROS

`counters [new_mon new_det]` – With scaler channel arguments, this macro assigns those channels to the MON and DET symbols and displays the changed scaler assignment.

Without arguments, the macro displays the current counter assignments, then prompts for new counter channels for MON and DET.

`ct` – A macro that counts for COUNT time if invoked without arguments, or for the time given as an argument. After counting, the macro prints the scaler contents. If interrupted with a ^C, the counts at the time of the interrupt are displayed.

`uct` – As above, but updates the screen with the scaler contents while counting. The screen is updated every UPDATE seconds. Only the first six counter channels are displayed.

`uctn` – As above, but uses cursor control commands to update a multiple line display at the bottom of the screen and will work with more than six counters.

`show_cnts` – A macro that reads counters and displays the scaler contents on the screen.

`scan_count` – Normally defined as `_count`, it is called by all the scan macros. It is redefined in powder mode to be `_pcount`.

`_count` – Normally defined as `count`, redefined by the `setscans` macro when updated counting during scans is in effect.

`count` – Default routine for counting during scans. It waits for moving to end, counts, waits and reads scalers. It also uses the `chk_beam` macro.

`chk_beam` – A *hook* macro, normally defined as `break`. Can be used to wait for sufficient counts during scans. See the macro source file `count.mac` for details.

`user_getcounts` – A *hook* macro that allows the user to insert code, if necessary, after the call to `getcounts` either to postprocess the the counter values, or to implement user-level counters. For example, in a configuration where the distance between the detector and the sample varies with the cosine of an angle, `user_getcounts` is defined to multiply  $S[\text{DET}]$  by an appropriate factor.

`user_precount` – A *hook* macro that allows the user to insert code to be run before the call to `tcount()` or `mcount()`.

`user_postcount` – A *hook* macro that allows the user to insert code to be run after the call to `tcount()` or `mcount()`.

A negative argument to `ct`, `uct`, `uctn` or `count` means count to monitor counts.

## SEE ALSO

`ct wait`

Hardware notes in the *spec Reference Manual*