

**NAME**

pca – Oxford/Tennelec/Nucleus PCA II, PCA-3, PCA Multiport MCA

**DESCRIPTION**

The PCA II and PCA-3 are ISA (PC card) multichannel analyzers. The PCA Multiport is a standalone MCA that `spec` currently supports using GPIB. The PCA-3 and PCA Multiport both support counting to live-time and real-time presets and both return dead-time information. The PCA II only supports presets and dead-time corrections when used in conjunction with a CSS provided driver, which is only available on selected PC UNIX platforms. Otherwise, the PCA II can be used in user-level I/O mode, with the timing performed by some other hardware timer or the software clock.

The PCA-3 and PCA Multiport come with 16384 channels, while the PCA II only supports up to 8192 channels. When acquiring data, the channels can be partitioned into groups of 256, 512, 1024, ... channels up to a single group that contains all the channels. Thus, the PCA-3 and PCA Multiport allows a maximum of 64 groups of 256 channels, 32 groups of 512 channels, etc. The maximum number of groups on the PCA II depends on the amount of memory installed on the board.

The PCA II in user-level mode is selected in the config file using

```
PC_PCAII = anything port_address POLL
```

The interrupt-driven mode is selected using

```
PC_PCAII = /dev/pca port_address INTR
```

See the *drivers/README* file in the `spec` distribution for information on installing the driver.

If using the interrupt-driven mode, note the following: Apparently, the PCA II doesn't trigger an interrupt on some PC mother boards. This problem can be fixed by changing the value of the resistor labeled R12 on the "PCA2 Memory Card" circuit diagram. This resistor is located near the lower left corner of the main board when viewed from the component side with the connector fingers pointing down and the input BNC to the right. R12 is about a centimeter down and to the left of the U26 IC. The circuit diagram indicates the resistor's value is 2K, however the boards seem to be shipped with a 1K resistor (brown-black-red stripes). Soldering a second 1K resistor alongside R12 and in parallel electrically will lower the resistance to 0.5K, which seems to work. (This modification was suggested by the manufacturer.)

**FUNCTIONS**

`mca_get(arr)` or `mca_get(g, e)` – Gets data from the currently selected MCA-type device, and transfers it to the array `arr` or to element `e` of data group `g`. Use of arrays rather than data groups is recommended. The native data type is `ulong`.

`mca_par("disable", arg)` – If `arg` is 1, prevents the PCA from being started and stopped by the standard counting functions `tcount()` and `mccount()`. If `arg` is 0, the PCA will be started and stopped with the standard counting functions (which is the default behavior).

`mca_par("clear")` – Clears the channels of the current group.

`mca_par("run")` – Programs the board with the current parameters and starts acquisition. For the PCA-3 and Multiport, acquisition time will be to the selected preset in real time or live time in PHA mode or the selected preset in number of passes in MCS mode. Note that the `tcount()` and `mccount()` functions, as used in the various counting macros, will also start PCA acquisition, however the PCA will free run until the count time passed to those functions has expired.

`mca_par("halt")` – Stops acquisition. Note that the PCA will also be halted when the `tcount()` and `mccount()` functions (as used in the various counting macros) complete

their count intervals or are aborted.

`mca_par("group_size")` – Returns the current group size.

`mca_par("group_size", size)` – Sets the group size to *size*. Legal values are 256, 512, 1024, 2048, 4096, 8192 and 16384. On the PCA II, values above 1024 may not be legal if insufficient memory is installed on the board.

`mca_par("select_group")` – Returns the currently active group. Groups are numbered starting at zero.

`mca_par("select_group", group)` – Set the active group to *group*. The maximum number of groups equals the number of channels divided by the group size. If the *group* passed to the function is greater than the maximum number of groups, the current group selected is *group* modulus the maximum number of groups.

`mca_par("pha")` – Selects pulse-height analysis mode.

`mca_par("gain")` – Returns the current gain value used in pulse-height analysis mode.

`mca_par("gain", value)` – Sets the pulse-height analysis gain to *value*. Legal values are 256, 512, 1024, 2048, 4096 and 8192.

`mca_par("offset")` – Returns the current digital conversion offset used in pulse-height analysis mode.

`mca_par("offset", value)` – Sets the pulse-height analysis digital conversion offset to *value*. Legal values are multiples of 256 from 0 to 7936. An offset value shifts the channel contents.

`mca_par("mcs")` – Selects multichannel scaling mode.

`mca_par("dwell")` – Returns the current multichannel scaling dwell time.

`mca_par("dwell", value)` – Set the multichannel scaling dwell time. Allowed values are numbers between 1e-6 and 60 seconds with mantissa of 1, 2, 4 or 8. A value of -1 selects external dwell. If *value* isn't an allowed value, it is rounded to the nearest allowed value.

`mca_par("mode")` – Returns two if the PCA is in PHA live-time mode, one if the PCA is in PHA real-time mode and zero if the PCA is in MCS mode.

`mca_par("readone", channel)` – Returns the contents of channel number *channel*. The channel number is with respect to the current group.

`mca_par("chan#")` – Returns the contents of channel number #. The channel number is with respect to the current group.

`mca_par("chan#", value)` – Sets channel # to *value*. The channel number is with respect to the current group.

The following `mca_par()` functions are valid for the PCA-3 and PCA Multiport devices, but are only valid for the PCA II when the board is used with the interrupt-driven driver.

`mca_par("preset")` – In PHA mode, returns the current live-time or real-time preset value in seconds.

`mca_par("preset", value)` – In PHA mode, sets the current live-time or real-time preset value to *value* seconds. The PCA counts to the preset using `mca_par("run")`. When using `mcoun()` or `tcoun()` the PCA acquires until halted by `spec`.

`mca_par("passes")` – In MCS mode, returns the number of preset passes.

`mca_par("passes", value)` – In MCS mode, sets the number of passes to *value*. When using `mcoun()` or `tcoun()` the PCA acquires until halted by `spec`.

mca\_par("live") – In PHA mode, selects live-time counting.

mca\_par("real") – In PHA mode, selects real-time counting.

mca\_par("dead") – In PHA mode, returns the percent dead time, if accumulating in live-time mode.

mca\_par("elapsed\_live") – In PHA live-time mode, returns the elapsed live time in seconds.

mca\_par("elapsed\_real") – In PHA mode, returns the elapsed real time in seconds.

mca\_par("elapsed\_passes") – In MCS mode, returns the elapsed number of passes.

**SEE ALSO**

wait