

**NAME**

epics – EPICS specific functions

**DESCRIPTION**

For versions of `spec` compiled for the EPICS environment (as used at the Advanced Photon Source and elsewhere), the three functions `epics_get()`, `epics_put()` and `epics_par()` provide generic *channel access* to EPICS process variables.

The EPICS version of `spec` also includes built-in support for the standard EPICS motor, scaler, MCA and generic-serial records which allows access to EPICS motors, scalers, MCAs and serial devices through `spec`'s hardware independent functions and commands.

**FUNCTIONS**

In the `epics_get()` and `epics_put()` functions below, note the special considerations given when the process variable is a byte array (EPICS type `DBF_CHAR`). The special treatment is given as these arrays are often used to hold strings since the maximum length of the EPICS `DBF_STRING` type is only 40 bytes. Many EPICS records use arrays of `DBF_CHAR` where longer strings are needed. (The special treatment for `DBF_CHAR` arrays is as of `spec` release 5.08.06-5.)

Note also that prior to EPICS 3.14, the maximum size of a data transfer was 16,384 bytes. With release 3.14 and later, for larger transfers, one can set the `EPICS_CA_MAX_ARRAY_BYTES` environment variable to larger values to overcome that limit.

The following functions for generic EPICS access are available:

`epics_get(ca_name [, type] [, count])` – Returns the result of a channel-access get of the process variable `ca_name`, as in `print epics_get("tmm:m1.VAL")`. By default, the native data type of the process variable will be retrieved over channel access. However, a different data type can be specified using the optional second argument. Possible types are "char", "short", "long", "float", "double", "enum" and "string". For `DBF_ENUM` data types, the string value of the process variable is returned, unless another type is specified. For array types, the entire array is retrieved and returned, unless the optional argument `count` specifies fewer elements (as of `spec` release 5.00.04).

When requesting "string" for types other than `DBF_CHAR`, arrays will be returned as a `spec` string data array consisting of 40-byte rows, with as many rows as elements in the EPICS array. Each row will contain an ASCII representation of the corresponding value, with the rest of the row padded with null bytes. For `DBF_CHAR` data, the return value will be a single-row string data array with as many columns as elements in the EPICS array.

`epics_put(ca_name, value [, wait_time])` – Does a channel-access put of `value`, which can be either a string or a number, to the process variable `ca_name`. If the optional `wait_time` argument is set, `spec` will wait for a channel-access "callback" to indicate the value has been sent. `spec` will wait up to the specified number of seconds for that response. Such a feature might come in handy if the channel-access put is to start an acquisition device and one wants to insure the device is started before continuing.

If the process variable is an array of a `DBF_CHAR` type, any non-array `value`, whether number or string, will be sent via channel access as a string, filling as many elements of the array as the string is long.

`epics_par(ca_name, what)` – Returns parameters associated with the process variable `ca_name`, according to the value of the string `what`, which may have the following values:

- "connect" - Queues the call to make the initial connection for the designated process variable, but doesn't force a network packet to be sent. If many process variables are to be accessed from user level, it will be more efficient in terms of network access to use this "connect" option for each, before the individual `epics_get()` and `epics_put()` calls. (Available as of `spec` release 5.08.01-4.)
- "count" - Returns the element count associated with `ca_name`.
- "host" - Returns the host name for `ca_name` (EPICS 3.12 and above).
- "monitor\_set" - Creates a "monitor" for the indicated process variable.
- "monitor\_check" - If the indicated process variable has a monitor, returns the value 1 if the value of the process variable has changed since last read with `epics_get()`. Returns -1 if no monitor has been created and for other EPICS errors. Otherwise returns zero.
- "monitor\_clear" - Removes the monitor for the indicated process variable.
- "read\_access" - Returns nonzero if `ca_name` has read access (EPICS 3.12 and above).
- "timeout" - Returns the value `spec` uses for the `ca_pend_io()` time-out parameter.
- "type" - Returns a string (such as "string", "double", "short", etc.) indicating the data type of `ca_name`.
- "write\_access" - Returns nonzero if `ca_name` has write access (EPICS 3.12 and above).

`epics_par(ca_name, what, value)` - Sets parameters associated with the process variable `ca_name`, according to the value of the string `what`, which may have the following values:

- "timeout" - Sets the value `spec` uses for the `ca_pend_io()` time-out parameter.

## ERRORS

When linked with the EPICS channel-access libraries, `spec` creates two built-in global variables, `EPICS_ERR` and `EPICS_ERR_MSG` (as of `spec` release 5.08.01-1). If one of the above functions results in an error generated by the channel-access calls, the value of the error will be assigned to `EPICS_ERR` and the corresponding message, as provided by the `ca_message()` function, will be assigned to `EPICS_ERR_MSG`. In addition, for certain errors at the `spec` level, including errors associated with failure to connect to an EPICS server, will result in a -1 assigned to `EPICS_ERR` and, in some cases, a message assigned to `EPICS_ERR_MSG`.

## MONITORS

To avoid excess network traffic, monitors can be created for selected process variables. With a monitor, `epics_get()` calls will return a cached value of the process variable, rather than calling out over the network to fetch a value. The EPICS database will update `spec` when the value of the process variable changes. The "monitor\_check" option to `epics_par()` allows for testing whether the value has changed since the last time the process variable was read with `epics_get()`.

Note, a `reconfig` (or `config`) call reinitializes the hardware connections and clears all monitors. Use the `config_mac` feature to set up macros to have monitors automatically created when hardware is initialized or reinitialized.

## DEFAULT TIMEOUT

The default timeout for the channel access calls is 0.5 seconds. The timeout for individual process variables can be changed using the `epics_par()` function, as described above. It also possible to change the default timeout for all newly created connections using the `spec_par()` function with the "epics\_timeout" option. (See the `spec_par` help file.) The parameters set with `spec_par()` are saved in the user's state file, so will be restored when `spec` is restarted, except, of course when starting fresh (with the `-f` flag).

## USING EPICS MOTORS

spec supports motors under EPICS using the standard EPICS motor record. spec provides two flavors of support. With the first (the EPICS\_M1 motors), spec takes all the motor parameters, such as step size, velocity, acceleration, etc., from spec's own configuration file. With the second (the EPICS\_M2 motors), the parameters are taken from the EPICS data base, and the parameters in the spec configuration file are ignored. Most sites configure the spec motors as EPICS\_M2 so that spec will not cause conflicts with other EPICS utilities.

Note, EPICS\_M1 motors write all the parameters configured in the spec configuration file to the EPICS data base when spec reads its configuration file, ignoring values currently set in the EPICS data base. However, spec will accept the position of the motor when first connecting to the data base, if the motor position is nonzero. If the motor position is zero and spec's settings file contains a nonzero position, spec will ask which position should be used.

For EPICS\_M2 motors, spec will only change motor parameters in the EPICS data base with the `motor_par()` function. (See the *motors* help file.)

Each set of motors with the same EPICS prefix should be entered on a separate line on the devices screen of the configuration editor:

```
MOTORS      DEVICE  ADDR  <>MODE  NUM          <>TYPE
   YES      ioc1:                44      EPICS Motor Controller
   YES      ioc2:                12      EPICS Motor Controller
```

The NUM field should be set to the highest channel number expected. On the motors screen of the configuration editor, motors are assigned by choosing EPICS\_M1 or EPICS\_M2 in the controller field, as in:

```
Number: <>Controller  0:EPICS_M2  1:EPICS_M2  2:EPICS_M2  3:EPICS_M2
Unit/Channel                0/1      0/2      0/3      0/4
Name                        Two Theta  Theta    Chi      Phi
Mnemonic                    tth      th      chi     phi
```

The optional unit/channel numbering is turned on for all EPICS motors by entering a unit/channel for any of the EPICS motors. The feature is disabled by typing ^D when the unit/channel data is highlighted for any of the EPICS motors. When the feature is disabled, channel numbers are assigned consecutively. Unit numbers are assigned according to the order the EPICS motor controllers appear on the devices screen, starting from zero. Note, though, channel numbers start at one, to match the APS EPICS motor record convention.

The APS EPICS motor naming convention appends m1, m2, etc., to the prefix defined on the motor controller screen.

An alternative naming scheme can be selected by entering a string value for the optional "Generic Parameter 1" on the second optional motor parameter screen. That string value will be appended to the prefix string defined on the motor controller screen.

With the following configuration:

```
MOTORS      DEVICE  ADDR  <>MODE  NUM          <>TYPE
   YES X04SA-ES2-                44      EPICS Motor Controller

Number: <>Controller  0:EPICS_M2  1:EPICS_M2  2:EPICS_M2  3:EPICS_M2
Unit/Channel                0/1      0/2      0/3      0/4
Name                        Two Theta  Theta    Chi      Phi
Mnemonic                    tth      th      chi     phi
...
Generic parameter 1          SAM:TTH    SAM:TH    SAM:CHI    SAM:PHI
```

The process variable prefixes would be X04SA-ES2-SAM:TTH, X04SA-ES2-SAM:TH, etc.