

**NAME**

taco/esrf – TACO (ESRF) specific functions

**DESCRIPTION**

spec provides two specialized functions to communicate with the TACO system of distributed servers and drivers developed at the ESRF for hardware control. The function `taco_io()` (or `esrf_io()`) provides generalized access to any control hardware that is supported by an ESRF driver. The function `taco_dc()` (or `esrf_dc()`) provides access to the ESRF data collector facility. The function `taco_db()` (or `esrf_db()`) allows read/write access to the TACO static resource database.

Support for specific TACO motor controllers (MAXE, VPAP) and counters (VCT6, VDL, CAEN 462, CAEN V560, LeCroy 1151) is built in to spec. These devices are normally accessed through spec's basic commands for moving and counting. spec also provides support for the TACO MCA and CCD device servers through the standard `mca_xxx()` and `image_xxx()` functions.

**HARDWARE ACCESS**

`taco_io(dev, cmd [, arg1_in [, arg2_in]] [, array_out][[, grp, ell])` – Allows general access to ESRF hardware device servers. The argument *dev* is the (case-insensitive) string name of the hardware device as specified in the corresponding device server documentation. The name might represent an entire VME module or it might represent an individual channel on a module. The argument *cmd* is the string name of a valid command for the device. Each command has associated specific input and output data types. At present, the following data types are supported.

D_VOID_TYPE	void
D_BOOLEAN_TYPE	Boolean
D_SHORT_TYPE	short
D_USHORT_TYPE	unsigned short
D_LONG_TYPE	long
D_ULONG_TYPE	unsigned long
D_FLOAT_TYPE	float
D_DOUBLE_TYPE	double
D_STRING_TYPE	string
D_VAR_CHARARR	char array
D_VAR_SHORTARR	short array
D_VAR_LONGARR	long array
D_VAR_ULONGARR	unsigned-long array
D_VAR_FLOATARR	float array
D_VAR_DOUBLEARR	double array

D_VAR_STRINGARR	string array
D_INT_FLOAT_TYPE	one long, one float
D_MOTOR_FLOAT	one long, one float
D_ATTE_TYPE	two shorts
D_LONG_READPOINT	two longs
D_MOTOR_LONG	two longs
D_FLOAT_READPOINT	two floats
D_DOUBLE_READPOINT	two doubles
D_STATE_FLOAT_READPOINT	one short, two floats
D_VAR_GRPFPARR	array of DevGrpFramePair objects

The `D_VOID_TYPE` data type means no data value is to be passed. `spec` treats all data as either a string or a double – the necessary type conversions are made automatically. The argument `arg1_in` is necessary if the command requires an input value. For the two-element composite types, `arg2_in` must also be provided. For array input types, `arg1_in` is the name of the array. For the `D_VAR_` array types (except for `D_VAR_STRINGARR`), `arg1_in` can be an associative array or a declared array data type. For commands that produce an output array or composite values, `array_out` is the name of an array to receive the values. The declared array data types can only be used for `D_VAR_` array types (except for `D_VAR_STRINGARR`). For commands that produce an output array of numbers (not strings or composite data types), a data group and element can be specified as `grp` and `el`, and the received data will be placed there.

`taco_io("?" [, filter])` - A single "?" lists all the devices available on the current `NETHOST`. The optional filter argument can be used to limit the contents of the returned list and is in the "usual" `DOMAIN/FAMILY/MEMBER` format, where the \* character can be used to match patterns in any of the three fields. See the `TACO db_getdevexp` man page for more information.

`taco_io(dev, "?")` - Lists the valid commands for the named device, along with the data types of any input and/or output arguments.

`taco_io(dev, "timeout", time)` - Sets the remote procedure call (RPC) timeout for the indicated device. The value of `time` is in seconds. If `time` is zero, the timeout is set to its original value. The function returns the timeout value.

`taco_io(dev, "udp"|"tcp")` - Sets the ethernet protocol for the named device.

If there is an error in sending a command to the device, `taco_io()` returns `-1`. In addition, the variable `TACO_ERR` (and `ESRF_ERR`, if it exists) will be assigned the device server error number (or zero if there is no error). If

TACO\_ERR (or ESRF\_ERR) has the value `-1` before the call of `taco_io()`, no error message will be printed, although, the error message will be assigned to the variable `TACO_ERR_MSG` (and `ESRF_ERR_MSG`, if it exists). If no output argument is generated by the command, the function returns zero. If the output data type is an array or composite value, the number of elements in the array or composite value is returned. Otherwise the value generated by the command is returned.

The first time `taco_io()` is called with a particular device name, the device is “imported” from the server. The connection remains open until you exit `spec` or execute the `reconfig` command. (The `reconfig` command is called by the `config` macro.) The `reconfig` command frees all server connections, so the next time `taco_io()` is called the device will be “imported” again.

The complex data type `D_VAR_GRPFPARR` is described as follows:

```

struct Frame {
    double  value;
    short   output, pause;
};
struct FramePair {
    Frame   dead, live;
};
struct DevGrpFramePair {
    u_int      nb_framepair;
    FramePair  framepair;
};
struct DevGrpFramePairArray {
    u_int      length;
    DevGrpFramePair *sequence;
};

```

with the `spec` array values assigned from the input array or to the output array in the order:

```

element[0] : sequence->nm_framepair
element[1] : sequence->framepair.dead.value
element[2] : sequence->framepair.dead.output
element[3] : sequence->framepair.dead.pause
element[4] : sequence->framepair.live.value
element[5] : sequence->framepair.live.output
element[6] : sequence->framepair.live.pause

```

ESRF devices named in the `config` file that are imported for motor, timer or counter control or for GPIB access can be accessed using `taco_io()`, although that is generally not recommended, as the user-level commands

might conflict with the internal code.

The string error message associated with any errors (either from `taco_io()` or from the built-in C code that accesses the device servers) will be assigned to the built-in variable `TACO_ERR_MSG` (and `ESRF_ERR_MSG`, if it exists).

## DATA COLLECTOR

The ESRF data collector system can be accessed using the following functions. All devices must reside on `NETHOST`.

`taco_dc(dev, "create", cmd, data_type)` – Creates a new pseudodevice for the data collector named `dev` having data accessed using the command `cmd` having data type `data_type`. The parameters `dev`, `cmd` and `data_type` are all strings. `spec` requires data collector devices it creates to have only one command. If the device already exists with a different command or with more than one command, the function fails. The `cmd` argument must be one of the commands recognized by the data collector. `spec` supports the following data types:

```
D_BOOLEAN_TYPE
D_SHORT_TYPE
D_LONG_TYPE
D_FLOAT_TYPE
D_DOUBLE_TYPE
D_STRING_TYPE
D_VAR_CHARARR
D_VAR_SHORTARR
D_VAR_LONGARR
D_VAR_ULONGARR
D_VAR_FLOATARR
D_VAR_DOUBLEARR
D_VAR_STRINGARR
```

`spec` treats all data as either a string or a double. The necessary type conversions will be made on the data passed through `taco_dc()`.

`taco_dc(dev, "delete")` – Deletes the device `dev` from the data collector.

`taco_dc(dev, "put", data)` – Writes data to the device `dev`. The device must exist in the data collector and must only have one command. The `data` argument will be converted to the appropriate string or number data type, according to the data type required by the device's command. If the command uses an array data type, `data` should be an array name.

`taco_dc(dev, cmd)` – Returns the string or number value associated with the command `cmd` for device `dev`.

`taco_dc(dev, cmd, array_name)` – If the command *cmd* for device *dev* returns an array data type, *array\_name* is the name of a SPEC array that will receive the string or number data. The function returns the number of elements in the array.

`taco_dc(dev, cmd, grp, el)` – If the command *cmd* for device *dev* returns an array data type containing number values, the arguments *grp* and *el* specify a SPEC data group and element number to receive the values. The function returns the number of elements in the array.

`taco_dc(dev, "?")` – Lists the commands and data types available for the device *dev*.

`taco_dc("?")` – Lists all the devices, commands and data types in the data collector for NETHOST.

If there is an error while executing `taco_dc()`, the function returns `-1`. In addition, if the error is associated with the data collector server, the data collector server error number will be assigned to the built-in variable `TACO_ERR` (and `ESRF_ERR`, if it exists). If no output value is generated by the function, it returns zero. If the output argument is an array, the return value is the number of elements in the array.

## RESOURCE DATABASE

`taco_db(dev, res)` – Returns a string containing the value of the resource *res* for the database item *dev*. The string *dev* is of the form `//facility/domain/family/member` or `domain/family/member`, with the default *facility* taken from the NETHOST environment variable. If there are multiple elements in the resource value, they are returned in a comma-separated list.

`taco_db(dev, res, value)` – The first two arguments are as above, with the resource *res* associated with *dev* being set to *value*. If multiple elements are required, the argument *value* should be a string in a comma-separated list.

If there is an error while executing `taco_db()`, the functions return `-1`. In addition, the variable `TACO_ERR` (and `ESRF_ERR`, if it exists), will be assigned the error number returned by the TACO library routines.