

NAME

AI Solutions DAQ-ATDC/NDAQ – USB acquisition module (formerly "KISIM")

DESCRIPTION

The NDAQ module was originally developed by the Korean Astronomy Observatory (KAO) group in collaboration with at the HANARO research reactor group at the Korean Atomic Energy Research Institute (KAERI). Development of the module has been taken over by the new company AI Solutions, which also created the DAQ-ATDC module. The NDAQ module (formerly called KISIM) requires an external time-to-digital converter (TDC), which is usually the ESRF N110 device. The DAQ-ATDC includes TDC functionality. Both modules communicate with the *spec* computer over a USB interface. Both USB 1.1 and 2.0 are supported.

The modules have several modes of operation. *spec* supports *static* mode in both linear and area detector configurations. Multiple modules can be used at the same time, with both 1D and 2D modes configured. Modules can also be configured with one master and several slaves, such that the master will gate the slaves to run at the same time.

The data returned by the DAQ-ATDC/NDAQ modules are four-byte long integers. Although the array size is configurable to 256, 512, 1024 and 2048 dimensions, the entire array must be received when reading the device. No area-of-interest feature is available. Using USB 2.0, *spec* can read a 1024×1024 array in about 1.75 seconds.

In 2D mode when acquisition is active, the device returns a 256×256 preview image no matter what the configured resolution.

CONFIGURATION

Select the modules on the **Acquisition** screen of the configuration editor.

MCA-like	DEVICE	ADDR	<>MODE	<>TYPE
0	YES	2	AI Solutions DAQ-ATDC/NDAQ	1D (USB)
0	YES	3	AI Solutions DAQ-ATDC (master)	1D (USB)
0	YES	4	AI Solutions DAQ-ATDC (slave)	1D (USB)

CCD-like	DEVICE	ADDR	<>MODE	<>TYPE
0	YES	5	AI Solutions DAQ-ATDC/NDAQ	2D (USB)
0	YES	6	AI Solutions DAQ-ATDC (master)	2D (USB)
0	YES	7	AI Solutions DAQ-ATDC (slave)	2D (USB)

The value in the ADDR field should match the device ID configured for the particular DAQ-ATDC/NDAQ module. Valid IDs are from 1 to 99. Although the devices IDs are programmable, they must be set outside of *spec* by the manufacturer or by software provided by the manufacturer.

When used in master-slave (or synchronized-start) mode, the modules are to be connected such that one module (the master) gates others (the slaves). There can be only one master, but there is no limit on the number of slaves.

Other than the limitation that there can only be one master, the other configuration options can be freely mixed in the same set up.

FUNCTIONS

When configured as a 1D MCA-type device, the DAQ-ATDC/NDAQ is controlled by *spec*'s MCA functions, such as `mca_get()` and `mca_par()`. See the *mca* help file for details. When configured as 2D device, the DAQ-ATDC/NDAQ is controlled by *spec*'s image functions, `image_get()` and `image_par()`.

The `mca_get()` (or `mca_sget()`) and `image_get()` functions retrieve data from the DAQ-ATDC/NDAQ module. The `mca_par()` (or `mca_spar()`) and `image_par()` functions control the module behavior as follows. The `sel` parameter is the MCA- or image-device number

from the *config* file. The functions `mca_get()` and `mca_par()` use the currently selected MCA device chosen by the `mca_sel()` function.

```
mca_get(arr)
mca_sget(sel, arr)
image_get(sel, arr) – Reads data into the array arr. The native data type is
    ulong.
```

In image mode, when acquisition is active, the device returns a 256×256 preview image. However, `image_get()` must still be called with an array dimensioned to the full resolution of the device. `spec` will place the preview image in the first 256 columns and rows of the array.

```
mca_par("reset")
mca_spar(sel, "reset")
image_par(sel, "reset") – Resets the USB endpoints and the DSP. Note, the
    modules are always reset during hardware initialization at start up and on a
    reconfig. This command shouldn't be needed during normal operation.
```

```
mca_par("mirror")
mca_spar(sel, "mirror")
image_par(sel, "mirror") – Returns one or zero, depending on whether mirror
    mode is set.
```

```
mca_par("mirror", value)
mca_spar(sel, "mirror", value)
image_par(sel, "mirror", value) – Turns mirror mode on or off depending on
    whether value is nonzero or zero. Mirror mode is only available in synchroni-
zied-start mode when one module is configured as master and others as
    slaves. In mirror mode, the slave modules are programmed with the same
    acquisition parameters as the master, and the slave presets are set to zero,
    which means the slaves will count as long as gated by the synchronization sig-
    nal from the master.
```

```
mca_par("update")
mca_spar(sel, "update")
image_par(sel, "update") – Updating the parameters on the DAQ-ATDC mod-
    ules (not the NDAQ) takes a couple of seconds. For maximum efficiency, spec
    only writes the parameters if one or more have changed. The parameters will
    be updated to the module, if necessary, before the modules are started. To
    avoid a delay when starting the modules, the "update" option can be sent to
    ensure the parameters are current. If operating in synchronized-start mode
    and the "mirror" parameter is set, the slave modules will be updated with the
    same operating parameters as the master.
```

```
mca_par("clear")
mca_spar(sel, "clear")
image_par(sel, "clear") – Clears the current data. If operating in synchronized-
    start mode with master and slaves, if called for the master, the slaves will be
    cleared also.
```

```
mca_par("run")
mca_spar(sel, "run")
image_par(sel, "run") – Starts data collection. If operating in synchronized-start
    mode with master and slaves, if called for the master, the slaves will be started
    also. In addition, if auto-clear (clear-on-start) mode is enabled, data on both
    master and slaves will be cleared.
```

`mca_par("halt")`
`mca_spar(sel, "halt")`
`image_par(sel, "halt")` – Halts data collection.

`mca_par("npts")`
`mca_spar(sel, "npts")` – Returns the current number of points in 1D configuration.

`mca_par("npts", value)`
`mca_spar(sel, "npts", value)` – Sets number of points in 1D configuration.
Legal values for the parameter are 256, 512, 1024 and 2048.

`image_par(sel, "resolution")` – Returns the the current resolution value in 2D configuration.

`image_par(sel, "resolution", value)` – Sets resolution in 2D configuration.
Legal values for the parameter are 256, 512, 1024 and 2048.

`image_par(sel, "bits")` – Returns the current value of the *image bits* parameter.

`image_par(sel, "bits", value)` – Sets the *image bits* parameter. This parameter is used to reduce the number of bits in the raw data arriving from the TDC input. A value of 12 results in a 4096×4096 array, a value of 11 results in a 2048×2048 array, and so on.

`mca_par("preset_mode")`
`mca_spar(sel, "preset_mode")`
`image_par(sel, "preset_mode")` – Returns "counts" or "time" according to the current preset mode.

`mca_par("preset_mode", value)`
`mca_spar(sel, "preset_mode", value)`
`image_par(sel, "preset_mode", value)` – Sets the preset mode to time if *value* is the string "time" or has the number value of zero. Sets the preset mode to counts if *value* is the string "counts" or has a nonzero number value.

`mca_par("preset")`
`mca_spar(sel, "preset")`
`image_par(sel, "preset")` – Returns the current preset value. The value is in seconds if the current preset mode is "time" or counts if the current preset mode is "counts".

`mca_par("preset", value)`
`mca_spar(sel, "preset", value)`
`image_par(sel, "preset", value)` – Set the preset value to the specified number of seconds if the current preset mode is "time" or to the specified number of counts if the current preset mode is "counts".

`mca_par("elapsed_time")`
`mca_spar(sel, "elapsed_time")`
`image_par(sel, "elapsed_time")` – Returns the elapsed time in seconds of the most recent data acquisition. The command can be used while acquisition is in progress.

`mca_par("elapsed_counts")`
`mca_spar(sel, "elapsed_counts")`
`image_par(sel, "elapsed_counts")` – Returns the elapsed counts of the most recent data acquisition. The command can be used while acquisition is in progress.

```

mca_par("offset_x")
mca_par(sel, "offset_x")
image_par(sel, "offset_x") – Returns the x-offset parameter for the ATDC module.

mca_par("offset_x", value)
mca_spar(sel, "offset_x", value)
image_par(sel, "offset_x", value) – Sets the x-offset parameter for the ATDC module to value.

image_par(sel, "offset_y") – Returns the y-offset parameter for the ATDC module, which is only useful in 2D mode.

image_par(sel, "offset_y", value) – Sets the y-offset parameter for the ATDC module to value, which is only useful in 2D mode.

mca_par("bin_time")
mca_par(sel, "bin_time")
image_par(sel, "bin_time") – Returns the ATDC bin time in picoseconds.

mca_par("bin_time", value)
mca_spar(sel, "bin_time", value)
image_par(sel, "bin_time", value) – Sets the ATDC bin time to value.
    Allowed values are 110, 120, 130, 140 and 150. spec will choose the value closest to the value passed as an argument.

mca_par("gate")
mca_par(sel, "gate")
image_par(sel, "gate") – Returns the gate value for the ATDC module in nanoseconds.

mca_par("gate", value)
mca_spar(sel, "gate", value)
image_par(sel, "gate", value) – Sets the ATDC gate to value, with units of nanoseconds. The valid range is from 100 to 500.

mca_par("emulation")
mca_spar(sel, "emulation")
image_par(sel, "emulation") – Returns zero or one to indicate whether emulation mode is on. In emulation mode, the DAQ-ATDC/NDAQ module generates an artificial data set.

mca_par("emulation", value)
mca_spar(sel, "emulation", value)
image_par(sel, "emulation", value) – Sets emulation mode on or off according to whether the argument is nonzero or zero.

mca_par("dump")
mca_spar(sel, "dump")
image_par(sel, "dump") – Displays most of the DAQ-ATDC/NDAQ parameters.
    This command can be used while acquisition is in progress.

```

USAGE

The following example shows how to take data in 2D mode and display the data using the 2D display program *onze* from the ESRF (and downloadable at certif.com/pub/util/onze).

```

{
    shared along array data[1024][1024]

    image_par(0, "resolution", 1024)

```

```
    image_par(0, "preset_mode", "time")
    image_par(0, "preset", 10)
    image_par(0, "run")
    wait()
    image_get(0, data)
}
```

To display the data, from a shell run *onze &*. After *spec* starts and the shared data array has been created, go to the *onze* file menu and choose “open spec”, then click on the name of the shared array.

SEE ALSO

mca

www.aisolutions.co.kr

http://www.esrf.eu/Infrastructure/Computing/Electronics