Module Interface Board - MIB

Service Port
Interface Control Document

Version 1.1.1

Elwood C. Downey, September 30, 2003

Introduction

This document describes the communication protocol over the Service Port, SP, on the Module Interface Board, MIB. The MIB is physically connected to the EVLA LAN on the one hand and to one or more pieces of electronic equipment on the other. The MIB functions as a uniform LAN interface for this disparate equipment.

The SP is used primarily by software processes to perform operational array activities. The same command syntax shall be available for engineering personnel via telnet from the Nucleus Shell connection. When connected in this manner the response syntax will be plain text. Software client processes are prohibited from using the Shell connection.

The MIB presents the electronics equipment to the SP as one or more logical Devices, each of which has one or more Monitor Points, MP, or Control Points, CP. MP are read-only and CP are write-only, although the MIB shadows each CP so it may be read back later. Each MP and CP have a value and a set of Attributes that expand upon certain auxiliary issues related to that value, such as allowed minimum and maximum.

A logical Device need not correspond to a physical device. The association between logical Devices on the Service Port and physical devices at the hardware level is performed by a Personality Object, PO, within the MIB which in turn depends upon the exact equipment connected to the MIB. The MIB itself will be one such logical Device for controlling and monitoring functionality wholly within the MIB.

For completeness, it should be mentioned there exists another LAN connection on the MIB called the Data Port. It transmits all MPs with broadcast UDP datagrams on a periodic basis for any process wishing to know these values. MPs have attributes which effect the rate at which they are transmitted. Details of the Data Port are described elsewhere.

See the attached diagram for a schematic depiction of the overall MIB software.

Connection

The MIB shall present UDP port 13001 for service port commands. All communication shall use ASCII lines. Each command shall fit within one UDP packet so no terminator is required.

The only two MIB commands shall be Get and Set. Get passively queries information from the MIB, Set actively makes changes on the MIB or to the equipment associated with the effected Device.

After receipt of a Get command the MIB shall always return one response containing the requested information. A response from a Set command shall be optional. Device, MP and CP names shall consist of alpha, digit and underscore characters only. Case shall not be significant. MP and CP names taken together shall be unique for one Device.

In the discussion that follows, the term Property is used when either MP or CP may be substituted.

Syntax Key

`abc`	literal
`<abc>`	variable
`[abc]`	optional
`{a|b|c}`	one of a set

Command Format -- Commands to the MIB

```
get {<device>|*}[.{<property>|*}[.{<attribute>|*}]] ...
```

The `get` command queries information about devices, properties and their attributes. These three components are collected into a triple using a period(`.`) as the separator. One or more triples may be present, each separated by one or more blanks. Each component may be specified as a name to indicate a particular instance or as an asterisk (`*`) to indicate all instances. If the device component is `*` the response will include all attributes for each device. If the property component is absent the response only includes device information. If the attribute component is absent the `val` attribute is used.

- Get the name and other general information of all logical devices: `get *`

```
<MIBResponse status="ok">
        <device name="device1" sn="13242" description="Wonder Device" />
        <device name="device2" sn="6567" description="Great Device" />
</MIBResponse>
```

- Get the value of all properties on all devices: `get *.*`

```
<MIBResponse status=”ok”>
        <device name=”device1” sn=”13242” description=”Wonder Device”>
                <monitor name=”mx” />
                <monitor name=”my”/>
                <control name=”cx”/>
                <control name=”cy”/>
        </device>
        <device name=”device2” sn=”6567” description=”Great Device” >
                <monitor name=”ma”/>
                <monitor name=”mb”/>
                <control name=”ca”/>
                <control name=”cb”/>
        </device>
</MIBResponse>
```

- Get the values of all properties on device1: `get device1.*`

```
<MIBResponse status=”ok”>
        <device name=”device1”>
                <monitor name=”mx” val=”10”/>
                <monitor name=”my” val=”20”/>
                <control name=”cx” val=”30”/>
                <control name=”cy” val=”40”/>
        </device>
</MIBResponse>
```

- Get the values of all the max attributes for all properties on device1: `get device1.*.max`

```
<MIBResponse status=”ok”>
        <device name=”device1”>
                <monitor name=”mx” max=”100”/>
                <monitor name=”my” max=”200”/>
                <control name=”cx” max=”300”/>
                <control name=”cy” max=”400”/>
        </device>
</MIBResponse>
```

- Get the value of one property on device1: `get device1.mx`

```
<MIBResponse status=”ok”>
        <device name=”device1”>
                <monitor name=”mx” val=”10”/>
        </device>
</MIBResponse>
```

- Get several different values and attributes in one command: `get device2.ma device2.ma.max device1.mb.min`

```
<MIBResponse status=”ok”>
        <device name=”device2”>
                <monitor name=”ma” val=”100” />
                <monitor name=”ma” max=”200” />
        </device>
        <device name=”device1”>
                <monitor name=”mb” min=”-10” />
        </device>
</MIBResponse>
```

`set[@<time>] [-v] <device>.<property>[.<attribute>]=<value> ...`

The `set` command instructs the MIB to install new values for device properties and their attributes. These three components are collected into a triple using a period(`.`) as the separator. A triple is assigned a value by following it with equals (=) then the value. Spaces are not allowed on either side of the equals. One or more triples and their assignments may be present, each separated by one or more blanks. If the attribute component is absent the `val` attribute is used.

The `set` command shall support an optional `-v` parameter. Without this parameter the `set` command never produces a response. With this parameter the response indicates whether the command was performed, as described elsewhere.

The `set` command shall support an optional time tag suffix to specify a future moment when the command should take effect. Without a time tag, `set` performs the actions at its earliest convenience and the response, if enabled with `-v`, refers to successful completion of the actions. When a time tag is specified, `set` queues the actions so they are performed at the specified time and the response, if enabled with `-v`, refers to successful queuing of the actions.

- Set several different values and attributes in one command: `set device2.ma.max=40  device1.mx=5`

  Confirm: `get device2.ma.max device1.mx`

```
<MIBResponse status="ok">
        <device name="device2">
                <monitor name="ma" max="40" />
        </device>
        <device name="device1">
                <monitor name="mx" val="5" />
        </device>
</MIBResponse>
```

`<time>`

The moment when this `get` or `set` is to occur. If absent perform at earliest convenience. There are two formats:

perform at given moment, ISO 8601 UTC:	`YYYY-MM-DDTHH:MM:SS.SS`
perform at given modified Julian date:	`52906.202948` (MJD = JD - 2400000.5)

`<property>`

Name of Monitor or Control point. These are unique to the equipment being controlled by the MIB.

The logical MIB Device shall support the following CP and so are available on all MIBs.

MIB CP:

`lock`	when first set to 1 MIB records the client IP. Advises Personality Object to screen subsequent `set` commands if they came from a different IP. remains in effect until set back to 0 by the same IP.
`flash`	when set to 1 MIB listens to TCP port 13002 for a new FLASH image in S Record format

`<attrib>`

All MP have the following attributes, however some may be ignored as appropriate:

`max`	maximum value, engineering units
`max_arm`	1 if an alarm is to be asserted if value exceeds `max`, else 0
`max_alarm`	1 if a `max` alarm is currently being asserted, else 0 (read-only)
`min`	minimum value, engineering units
`min_arm`	1 if an alarm is to be asserted if value falls below `min`, else 0
`min_alarm`	1 if a `min` alarm is currently being asserted, else 0 (read-only)
`val`	current value, engineering units (default attribute)
`aperiod`	period between archive broadcasts on data port, ms
`operiod`	period between observing broadcasts on data port, ms
`speriod`	period between screen broadcasts on data port, ms
`slope`	along with `intercept` determines how raw units are converted to engineering units*
`intercept`	along with `slope` determines how raw units are converted to engineering units*
`raw`	`val` as presented directly by the Module hardware, before conversion to engineering units

* The default transform from raw to Engineering units is Engineering unit = `raw * slope + intercept` but a Personality Object is free to ignore these values and perform its own transform if necessary. In the latter case, the PO will report an error if `slope` or `intercept` are referenced in a `set` or `get` command.

CP have the following attributes, however some may be ignored as appropriate:

`max`	maximum value, engineering units
`min`	minimum value, engineering units
`lastset`	time `val` was last set, read-only, MJD.
`val`	new value when written, last value `set` when read; engineering units (default attribute)
`slope`	along with `intercept` determines how engineering units are converted to raw units*
`intercept`	along with `slope` determines how engineering units are converted to raw units*
`raw`	value as presented directly to the Module hardware, without conversion from engineering units

* The default transform from Engineering to raw units is `raw = Engineering * slope + intercept` but a Personality Object is free to ignore these values and perform its own transform if necessary. In the latter case, the PO will report an error if `slope` or `intercept` are referenced in a `set` or `get` command.

`<value>`

New value for Property attribute. Only numeric values are supported except an asterisk (`*`) indicates set to a default value.

Response Format -- Commands from the MIB

All commands are immediately checked for validity before any action is taken. This checking shall include at least syntax errors; unknown device, property or attribute names; and values out of range. Failure at this step shall result in no change in status to the MIB or any connected equipment modules. Failures are always reported by the `get` command and optionally by the `set` command if the `-v` option was used.

All responses, if sent, are in XML format. The outer element shall be `MIBResponse` with one attribute, `status`. If the command was successful the value of the status attribute value shall be `ok`. If the command was `get` then additional subelements will be present within the `MIBResponse` element to report the queried values. See the discussion for the `get` command for examples of successful responses. If the command could not be performed for some reason the status attribute value shall be `err` and the text content of the `MIBResponse` element shall be a brief explanatory message. Some examples of failure messages:

```
get device3^ma
<MIBResponse status=”err”>Syntax error near: ^</MIBResponse>

get device3:ma
<MIBResponse status=”err”>Unknown device: device3</MIBResponse>
```

MIB Software Design

Version 0.1

September 26, 2003
Elwood C. Downey