Valmet DNA Engineering

Debugger Manual

Collection 2020 rev. 6

G5062_EN_06

Valmet Automation Inc.
P.O. Box 237, FIN−33101 Tampere, Finland
Tel. +358 (0)10 672 0000
www.valmet.com/automation
Documentation feedback: aut_manuals@valmet.com

Document History

Date	Revision	Comment
15.03.2021	**6**	**Valmet DNA Collection 2020** **No changes in contents.**
26.11.2019	**5**	**Valmet DNA Collection 2019** **No changes in contents.**
11.01.2019	**4**	**Valmet DNA Collection 2018** **No changes in contents.**
13.10.2017	**3**	**Valmet DNA Collection 2017** **No changes in contents.**
25.10.2016	**2**	**Valmet DNA Collection 2016** − **Removed some chapters related to VME nodes**
30.06.2015	**1**	**Valmet DNA Collection 2015**

Contents

1 General

1.1 Operation

The Diagnostics Server (DIA) Debugger tool has been designed for examining Valmet DNA and the internal workings of applications.

Typical applications of this tool are:

- testing of applications
- debugging applications
- analyzing the operation of Valmet DNA
- debugging Valmet DNA

With the Debugger tool you can examine and modify the internal data of Valmet DNA. Therefore you must have deep knowledge of how Valmet DNA works.

Chapter 6 ”Diagnostics Sensors” introduces some general Valmet DNA diagnostics sensors, which are used, for instance, in building system alarms. Using the Debugger tool you can examine the states and values of the diagnostics sensors.

Programs and software which handle one function of the Application Server’s whole functionality are referred to as components in this document. A component can consist of other components. For example, the PCS includes CPU and FBC components.

NOTE!
The Debugger tool includes commands that can not be executed in all kind of node types. For example serial interface control commands can be directed only to VME node.

2 Equipment and Its Installation

2.1 DIA−PC Node

2.1.1 The Structure of the DIA−PC Node

The DIA−PC node consists of a Windows based PC with its bus, mouse and keyboard ports. Note that the DIA component has usually been located in the same PC node with OPS or EAS components.

Engineering hardware (EAS and EAC), which is used for not only application engineering but can also be used to operate the debugger tool, is used as the diagnostic terminal. Engineering hadware communicates with the DIA−PC node through Ethernet. A common net printer is used as the printer.

3 Starting the Debugger Tool

3.1 Logging−in

3.1.1 DIA−PC Node

The Debugger is started at the engineering server (EAS) by Valmet DNA Program Starter ’slot’ command e.g. ’slot 10 start’. Anyhow by default the DIA is configured to start automatically according to the following example:

```
// DIA Debugger

Slot 10
set XD_STDIO=CONSOLE         // Open console window for the program
                             // in this slot.
set XD_DISK=                 // Use configuration files from
                             // DIA’s working dir.
set DIA_WORKING_DIR=.
set XD_IGNORE_SIGNALS=OFF    // get signals working.
                             // (ie.:  ctrl-c)
PRIORITY=HIGH
AUTOSTART=YES                // Start program automatically.
TITLE=AD00                   // Title of the program EDIT!
PROGRAM=D:/dna/CA/dia/dmng_6.19.exe  // CHECK path!
PARAMETERS=-ntcpu AD00_4.1_cpu     // EDIT correct cpu-file name!
BACKUPDIRS=d:/dna/CA/dia           // Backup the complete dia
                                   // directory.
```

When the Debugger tool is ready to be used, its window will be opened on the display with the following prompt on it:

```
1a%
```

Enter log(out) to close the Debugger tool.

3.2 Remote Telnet Connection to Debugger

The Debugger tool of the Diagnostics Server (DIA) can also be used outside the engineering hardware via a remote connection.

Note that only one user at a time can use the Debugger tool of the Diagnostics Server. Terminal user (see chapter 3.1 ”Logging−in”) is logged out for the duration of the telnet connection. Terminal user’s connection is restored when the telnet connection is closed with the command:

```
dis(connect)
```

The session of another active user can be closed by answering ’y’ to the question below:

```
*** WARNING WARNING WARNING ***

Active connection exists.
Do you want to override current user?
(please answer y(es)/n(o))
```

Answer ’n’ if the other session is not wished to be closed.

3.2.1 DIA−PC Node

The Debugger tool of the DIA−PC node can be connected remotely using the telnet protocol by giving the command

```
telnet <name or IP address of DIA-PC node> <port>
```

in which <port> is the default port 1234. Note that the Debugger tool must be started in the target DIA−PC node before opening the telnet connection. If this is not the case, see starting instructions in chapter 3.1.1 ”DIA−PC Node”.

When the Debugger tool is ready for use, the following prompt will appear:

```
1a%
```

Enter dis(connect) to close the telnet connection.

3.3 Modes

The Debugger tool has two operation modes:

- application mode
- system mode

After log−in the Debugger tool is in application mode. You can toggle the mode with the command

```
togglemode
```

The application mode and the system mode have different sets of commands (see the command charts).

4 Editor Functions

4.1 General Syntax of the Commands

A command consists of three parts:

- command words
- parameters
- options

The **ctrl−c** command allows you to stop executing the command activated last as a background process. All commands running as background processes can be stopped with **ctrl−c ctrl−c**. This feature is useful in, for instance, long printing jobs when you have already obtained the data you were looking for.

The prompt of the DIA−PC node differs from the prompt of the DIA−VME node in showing the mode (s/a) in use. For example, the prompt of the application mode is as follows:

```
1a%
```

4.1.1 Command Words

Command words tell Valmet DNA what to do. One command line may contain one or several command words. In most cases you only need the first character of the command word. Note that command words of the same level beginning with the same character form an exception.

For instance, the command

```
5s% print structure  <CR>
```

can be abbreviated as follows:

```
6s% p s  <CR>
```

No distinction is made between uppercase and lowercase characters. To expand a command word in the display press **ctrl−y** at any point of typing the command word.

4.1.2 Parameters

Parameters indicate the object of the command. Unlike command words, parameters cannot be abbreviated, you must enter them in full to eliminate possible ambiguity.

For example

```
5s% print structure <ncuid>  <CR>
```

In this command the parameter <ncuid> is the NCU’s hw address (you must enter it in full because there are several NCUs in Valmet DNA).

4.1.3 Options

Options govern the execution of commands. For example, you can define a command to be executed repeatedly at intervals of 5 s.

An option is the last member of the command line before **<CR>**. An option is one character preceded by a − character. One command may contain several options and the options can be given sequentially in any order or separated with spaces.

An option may require a parameter, which is inserted after the option. The parameter has a on default value, which is used if the parameter is not specified.

The following six options are available:

−l	request from default component
−r (<timeinterval>)	repeat command at specified intervals
−b	command runs on the background
−n	not displayed
−p <printer>	output to the printer
−i	continuous output without page breaks

Option −l

Option **−l** specifies that the data will only be requested from the default component, which speeds up the command execution. Using the default component is necessary, for instance, when the command is addressed to one Operation Server, because the same name may exist in several Operation Servers.

The default component is specified using the command

```
5a% set local <localcomponent>  <CR>
```

Option −r (<timeinterval>)

Option **−r** specifies a command to be executed at given intervals. The interval is given in seconds at an accuracy of 1 second. If the time interval is omitted, it will default to a previously used interval. If no previously used interval exists, the command will be executed at intervals of 10 s.

Option **−r** also contains the **−b** option (see the following option).

The use of this option is not allowed in connection with Modify commands.

To cancel repeating press **ctrl−c**.

Option −b

Option **−b** specifies the command to be executed on the background. During the execution of a background command other commands can be run on the foreground.

The use of this option is not allowed in connection with Modify commands.

A background command is cancelled by **ctrl−c ctrl−c**. This cancels all background commands at the same time.

To get a list of currently running background commands use the command:

```
5a% print background  <CR>
```

Option −n

Option **−n** prevents the output from being displayed on the screen. This option is useful, for example, with background printing.

The use of this option is not allowed in connection with Modify commands.

Option −p <printer>

To direct the output of a single command to the printer use option **−p**. This option must be followed by the printer name. The following command sends the name value to printer at intervals of 25 seconds, but not to the display.

```
5a% print variable :e:pr:FIC−100:me −r 25 n p <CR>
```

Option −i

Option **−i** specifies that the printing is done continuously, without page breaks (i.e. “−−More−− (q=quit)” notifications). This option is useful when outputting variable values with scripts.

```
5a% print variable :e:di:AP01:activity –i <CR>
```

4.2 Editing of Commands

You can expand a command word at any point of typing it by pressing **ctrl−y**.

When you have typed the command line or part of it (before ending it with **<CR>**), you can change it with editor commands. You can move the cursor along the line and make changes, deletions or additions.

The editor is either in Insert or Replace mode. In Insert mode you can type new text on the line inserting it between the character under cursor and the character before it. In Replace mode the new text replaces the text under it. After power−up the Diagnostics Server is in Replace mode. You can toggle between the modes with **ctrl−a**.

To get help on editor commands or to check current editor mode (Replace/Insert) press **ctrl−k**.

In addition the following commands are available:

MOVING COMMANDS:

ctrl−b	moves cursor backward one character
ctrl−f	moves cursor forward one character
ctrl−g	moves cursor to the beginning of the command line
ctrl−e	moves cursor to the end of the command line

DELETING COMMANDS:

ctrl−d	deletes character under cursor
DEL	deletes character under cursor
BS	deletes character before cursor

4.3 Default Text

The command

```
5a% set prefix <defaulttext>  <CR>
```

can be used for specifying a default text for the tool. When you want to include the default text in a command, you can type the * character instead of the text. The command word itself can however not be replaced by the default text.

Example

```
5a% set prefix :e:pr:FIC–100:  <CR>

   Default text:  :e:pr:FIC–100:

6a% print variable *me  <CR>
```

Command 6a% prints the value of the name ’:e:pr:FIC−100:me’.

4.4 Command History

20 last executed command are stored in chronological order in the command history.

Command history is displayed by the command **h!**.

Here is an example of a displayed history:

```
 ..  ...
 ..  ...
45 p v :e:pr:50–FIC–245:me –r 20
46 print structure
```

The commands in the history are numbered in ascending order starting from 1. Any command from the 20 last one can be displayed in the command line using the command

```
n! , where n is the number of the command.
```

The last executed command is displayed by **!** .

After calling a command from the command history to the command line you can edit it using the editor commands and execute it with a **<CR>**.

4.5 Help on Command Syntax

If you press **ctrl−x** or **?** at a command word, parameter or option in the command line, you will get a list of possible entries.

After displaying the help text Valmet DNA will rewrite your command line, so you can continue from the point where you left. You can now enter the desired alternative, and, if necessary, use the **ctrl−x**:n or **?**:n again for the next command, until you are ready to finish the line with a **<CR>**.

Example (in application mode):

```
5a%  print <ctrl-x>

   THE ALTERNATIVES AT THIS POINT ARE:

           background
           changes
           directory
           extensions
           module
           variable

5a%  print _
```

4.6 Summary of Editor Commands

ctrl−c	cancels currently executed command
ctrl−c ctrl−c	cancels the execution of all background commands
ctrl−y	expands the command word on the command line
ctrl−a	toggles between Insert/Replace modes
ctrl−k	help on editor commands, also shows editor mode (Insert or Re-place)
ctrl−b	moves cursor backward one character
ctrl−f	moves cursor forward one character
ctrl−g	moves cursor to the beginning of the command line
ctrl−e	moves cursor to the end of the command line
ctrl−d	deletes character under cursor
ctrl−x	help on commands
?	help on commands
DEL	deletes character under cursor
BS	deletes character before cursor
h!	command history (20)
n!	the nth command of command history
!	last executed command
ctrl−t	toggles the function mode application mode/system mode
ctrl−l	lists the name tree in the cursor location

5 Command Charts and Descriptions of Commands

5.1 Application Mode Command Chart

```
logout

modify ──── variable <name>

print ───┬── backround
         ├── changes <name> (rise/lower) (mask <value>) (<name1>) (<name2>)...
         ├── directory <dir>
         ├── extensions <name>
         ├── module ─┬── administration ─┬── <name>
         │           ├── local ──────────┤
         │           ├── function ───────┤
         │           ├── interface ──────┤
         │           ├── global ─────────┤
         │           ├── connection ─────┤
         │           ├── external ───────┤
         │           ├── data ───────────┤
         │           ├── name ───────────┤
         │           ├── specifier ──────┤
         │           └───────────────────┘
         └── variable <name1> <name2> ...

set ─────┬── local  <localcomponent>
         ├── prefix <defaulttext>
         ├── printstyle ─┬── traditional
         │               └── compact
         ├── printdepth <struct depth>
         └── printmaxdim <max dimension>

togglemode

systemmode

applmode

disconnect

lock

unlock

capture ─┬── <filename>
         └── off

script ──┬── <filename>
         └── off

sleep ────── <delay [ms]>
```

5.2 Description of Application Mode Commands

5.2.1 logout

logout

Resets the Diagnostics Server. Do not use this command for exiting the Debugger tool; use the disconnect command instead.

5.2.2 modify

modify variable <name>

To modify the contents of the specified name. The program displays the current contents and waits for you to enter the new contents. If you do not want to change the contents, press <CR>.

5.2.3 print

print backround

Prints all currently running commands.

print changes <name> (rise/lower) (mask <value>) (<name1>) (<name2>) (<name3>) ...

This command activates the monitoring of changes in the state of a binary signal. The command always runs in the background. The command's parameters are as follows:

<name>
This field specifies the signal to be monitored. The type of the signal must be 'bin' or 'binev'.

rise/lower
This field defines how the binary signal's status (bit 0) is actually to be monitored: either rising edges (rise) or falling edges (lower). If you omit this parameter, the command will monitor both edges. All changes of the other bits (= fault bits) will be monitored.

mask <value>
This field defines the bits to be monitored. You can mask off any bits you do not want to monitor. In the masks's <value> the bit state "1" means 'print the changes', while "0" means 'do not print the changes'. You give the <value> as a hexadecimal number.

<name><name2>...
These fields are extra names of variables. The variables given in these fields will always be printed in connection with the monitored signal. The variables can be of any type. The number of variables to be printed is not limited.

options
The normal options are available. When the command has been activated, the monitored signal's current state is printed first, after which the command waits for changes in the state. If the −r option is not used, the command will stop at the first printed change. If the −r option is used, the command will print the signal's state at each change until you stop the command.

print directory <dir>

Prints the contents of the specified name server directory.

print extensions <name>

Prints the type and specifiers of the specified name.

print variable <name1> <name2> <name3> ...

This command prints the type and specifiers as well as the value for the specified name. The command allows you to print the values of several signals at the same time. The number of signals to be printed is not limited.

5.2.4 print module

print module administration <name>

Prints the administration part of the specified module.

print module local <name>

Prints the local data points of the specified module.

print module function <name>

Prints the function part of the specified module.

print module interface <name>

Prints the interface ports of the specified module.

print module global <name>

Prints the direct access ports of the specified module.

print module connection <name>

Prints the connection area of the specified module.

print module external <name>

Prints the external data points of the specified module.

print module data <name>

Prints the data area of the specified module.

print module name <name>

Prints the name−ASCII area of the specified module.

print module specifier <name>

Prints a part list of the specified module.

print module <name>

Prints the specified module in full.

5.2.5 set

set local <localcomponent>

Sets default component.

When the default component has been set, subsequent −l options in command retrieve data only from the default component, not from the entire Valmet DNA.

Example 1:

```
5a% set local AP01  <CR>
```

This command defines Process Control Server AP01 as the default component.

Example 2:

```
7a% s l :e:pr:FIC−100:  <CR>
```

This command defines as the default component the Process Control Server where the name ':e:pr:FIC−100' exists.

set prefix <defaulttext>

Sets default text.

When you want to include the default text in a command, you can use the * character instead of the text. However, the command word itself cannot be replaced by the *.

Example

```
5a% set prefix :e:pr:FIC−100:  <CR>
       Default text :e:pr:FIC−100:
6a% print variable *me  <CR>
```

Command 6a% prints the contents of the name pr:FIC−100:me.

set printstyle traditional

Sets the traditional printing mode, in which structured types are printed one member at a time.

Printing example:

```
16a% p v :e:di:AD01:activity:fault
      Print Variable

   Lid AD01 found.
IS binev
   MEMBER IS binstat IS bin IS uns16 <0><0x0>
   MEMBER IS bintime IS time
      MEMBER IS hundus IS uns8 <112><0x70>
      MEMBER IS tenms IS uns8 <81><0x51>
      MEMBER IS sec IS uns8 <67><0x43>
      MEMBER IS min IS uns8 <0><0x0>
      MEMBER IS hour IS uns8 <6><0x6>
      MEMBER IS wday IS uns8 <5><0x5>
      MEMBER IS day IS uns8 <36><0x24>
      MEMBER IS month IS uns8 <17><0x11>
      MEMBER IS year IS uns8 <0><0x0>
      MEMBER IS vers IS uns8 <0><0x0>
```

set printstyle compact

Sets the compact printing mode, in which structured types are printed in a more compact format.

Printing example:

```
18a% p v :e:di:AD01:activity:fault
      Print Variable

    Lid AD01 found.
IS binev
    MEMBER IS binstat IS bin IS uns16 <0><0x0>
    MEMBER IS bintime IS time <Fri Nov 24 06:00:43.517 2000 v00>
```

set printdepth <struct depth>

Sets printing depth for structured types. The default value is zero, in which case structures are printed in full.

Printing example:

```
13a% p v :e:di:AP01:slot02:iorack00

      Print Variable

   Lid AP01 found.
IS dirk
   MEMBER IS type IS uns16 <1><0x1>
   MEMBER IS version(8) IS char “6.01”
   MEMBER IS fault IS binev
     MEMBER IS binstat IS bin IS uns16 <0><0x0>
     MEMBER IS bintime IS time
       MEMBER IS hundus IS uns8 <0><0x0>
       MEMBER IS tenms IS uns8 <0><0x0>
       MEMBER IS sec IS uns8 <0><0x0>
       MEMBER IS min IS uns8 <0><0x0>
       MEMBER IS hour IS uns8 <0><0x0>
       MEMBER IS wday IS uns8 <0><0x0>
       MEMBER IS day IS uns8 <0><0x0>
       MEMBER IS month IS uns8 <0><0x0>
       MEMBER IS year IS uns8 <0><0x0>
       MEMBER IS vers IS uns8 <0><0x0>
```

```
15a% set printde 2

  Print depth = 2

16a% p v :e:di:AP01:slot02:iorack00

      Print Variable

   Lid AP01 found.
IS dirk
   MEMBER IS type IS uns16
   MEMBER IS version(8) IS char
   MEMBER IS fault IS binev
   MEMBER IS faultind IS intsev
   MEMBER IS iocard(16) IS iocard
   MEMBER IS faultsum IS bin IS uns16
```

set printmaxdim <max dimension>

Sets a limit for printing the elements of the table types. For example, it is possible to set a limit to a print only elements 1−10 of a large table.

Printing example:

```
15a% set printm 10

  Print max dimension = 10

18a% p v :e:pr:ANA128X128.F:out1:elem
      Print Variable

   Lid AP01 found.
(16384) IS ana
(1)
   MEMBER IS f IS fails IS uns16 <32><0x20>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(2)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(3)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(4)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(5)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(6)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(7)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(8)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(9)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(10)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
(16384)
   MEMBER IS f IS fails IS uns16 <0><0x0>
   MEMBER IS a IS float <0.000> <+.00000000+00>
```

5.2.6 togglemode

togglemode

Toggles between application mode and system mode.

5.2.7 systemmode

Systemmode commands switches Debugger to system mode.

5.2.8 applmode

Applmode commands switches Debugger to application mode.

5.2.9 disconnect

disconnect

Exits the Debugger tool.

5.2.10 lock

lock

Locks a session preventing it to be aborted by another user.

5.2.11 unlock

unlock

Unlocks a session.

5.2.12 capture

capture <filename>

The capture command stores the user’s commands and their printouts into a file. Storing is activated by giving command ’capture <filename>’ and deactivated by command ’capture off’. The storing file can be directed to a local hard disk (debugger in a PC node) or a Backup Server.

Examples:

```
capture c:\temp\my_capture.txt
capture :s:AB01:my_capture.txt
```

If the file name does not include the path, for example ”c:\temp”, the storing file is created in the same place where the Debugger tool has been launched from (in a PC node the default is D:\dna\ca\dia).

capture off

Capture off commad deactivates storing the printouts into the file and cloces storing file. Stored file can be viewed only when file has been closed with the capture off command.

5.2.13 script

script <filename>

The script command reads command from a file and executes them. Script command is given in the form of ’script <filename>’ where the ’filename’ is the file’s name which contains the commands. The executed commands can be any kind normal Debugger commands.

An example of a file run with the Script command, where:

- printing mode is set as ’compact’
- the commands and printouts are directed to bu server’s file ’TX_out.txt’
- the diagnostics sensor member is listed twice with a 10 second delay in between

```
set printsty compact
capture :s:AB01:TX_out.txt
applmode
p v :e:di:AP01:node06:chan1:tx_count
sleep 10000
p v :e:di:AP01:node06:chan1:tx_count
cap off
```

5.2.14 sleep

sleep <delay [ms]>

The sleep command causes a delay, during which the Debugger does not handle commands. Sleep is used to create delays between commands in command scripts, which are run with the script command. The length of the delay is given in milliseconds, for example the command

```
sleep 10000
```

causes a 10 second delay.

5.3 System Mode Command Chart

```
clear ───┬─ eventhistory
         └─ from ────┬─ dcu ───┬─ addr <dcuid>
                     │         ├─ counters <dcuid>
                     │         ├─ servicetime <dcuid>
                     │         ├─ status <dcuid>
                     │         └─ traffic <dcuid>
                     ├─ fbc ───┬─ counters <fbcid>
                     │         ├─ io−diag <fbcid> <pic>
                     │         ├─ pic−counters <fbcid> <pic>
                     │         ├─ rupi−counters <fbcid>
                     │         └─ servicetime <fbcid>
                     ├─ ncu ──── eventhistory <ncuid>
                     └─ rsu ──── counters <rsuid>

kill <procid>

logout

modify ──┬─ memory ──┬─ cpu ───┬─ byte <addr>  ───┬─ (<cpuid>)
         │           │         ├─ word <addr>  ───┤
         │           │         └─ long <addr>  ───┘
         │           ├─ io <fbcid> <pic> <unit> < page> <addr> <count> <value>
         │           └─ pic ───┬─ external <fbcid> <pic> <addr> (<value>) (<value>) ...
         │                     └─ internal <fbcid> <pic> <addr> (<value>) (<value>) ...
         ├─ line ────┬─ baudrate <value> <linename>
         │           ├─ flowcontrol <value> <linename>
         │           └─ mode <value> <linename>
         └─ pic ─────┬─ counters <fbcid> <pic> <index> (<value>) (<value>) ...
                     ├─ global <fbcid> <pic> <index> (<value>) (<value>) ...
                     └─ state <fbcid> <pic> <index> (<value>) (<value>) ...

print ─┬─ backround
       ├─ directory ──── name ── <path> (<cpuid>)
       ├─ eventhistory
       ├─ grouplist (<groupid>)   Note! Available only in DIA−VME node
       ├─ line <linename>
       ├─ memory <addr> (<cpuid>)
       ├─ structure <ncuid>
       ├─ time
       ├─ recursive ncu_time <ncuid> <n>
       └─ from ── cpu ──┬─ load (<cpuid>)
                        ├─ machineinfo (<cpuid>)
                        ├─ peripherals (<cpuid>)
                        ├─ vme (<cpuid>)
                        ├─ time <cpuid>
                        └─ network delay <target cpuid> <source cpuid>   Note! Available only in DIA−VME node.
```

```
print ─── from ─┬─ dcu ──┬─ allgroups <dcuid>
                │        ├─ blockmessage <dcuid> <bus> <station> <message>
                │        ├─ classicbus <dcuid>
                │        ├─ errors <dcuid>
                │        ├─ group <dcuid> <group>
                │        ├─ history <dcuid>
                │        ├─ io−modules <dcuid>
                │        ├─ load <dcuid>
                │        ├─ moduledata <dcuid> <addr>
                │        ├─ servicetime <dcuid>
                │        ├─ stations <dcuid>
                │        ├─ version <dcuid>
                │        ├─ traffic <dcuid>
                ├─ fbc ──┼─ calculation ─┬─ allgroups <fbcid>
                │        │               ├─ counters <fbcid>
                │        │               ├─ fbcstates <fbcid>
                │        │               ├─ group <fbcid> <groupid>
                │        │               ├─ history <fbcid>
                │        │               ├─ io−module <fbcid>
                │        │               ├─ rolehistory <fbcid>
                │        │               └─ servicetime <fbcid>
                │        ├─ io ──────────┬─ diagnostic <fbcid> <pic>
                │        │               ├─ memory <fbcid> <pic> <unit> <page> <count>
                │        │               ├─ page <fbcid> <pic> <page> <count>
                │        │               ├─ structure <fbcid>
                │        │               └─ version <fbcid>
                │        ├─ pic ─────────┬─ counters <fbcid> <pic> (<index> <count>)
                │        │               ├─ global <fbcid> <pic> (<index> <count>)
                │        │               ├─ memory ─┬─ external <fbcid> <pic> <addr> (<count>)
                │        │               │          ├─ internal <fbcid> <pic> <addr> (<count>)
                │        │               │          └─ rom <fbcid> <pic> <addr> (<count>)
                │        │               ├─ state <fbcid> <pic> (<index> <count>)
                │        │               └─ version <fbcid>
                │        ├─ load <fbcid>
                │        └─ rupi ────────── counters <fbcid>
                ├─ fbc ──── traffic ─┬─ counters <fbcid>
                │                    ├─ history <fbcid>
                │                    └─ load <fbcid> <timeinterval>
                └─ kernel ─┬─ counters (<cpuid>)
                           ├─ descriptor <procid>
                           ├─ eventmask (<cpuid>)
                           ├─ groupidlist (<cpuid>)
                           ├─ information (<cpuid>)
                           ├─ logicalidlist (<cpuid>)
                           ├─ memory (<cpuid>)
                           ├─ processlist (<cpuid>)
                           ├─ status (<cpuid>)
                           └─ teamstatus <procid>
```

```
print ──── from ─┬─ ncu ──┬─ counters <ncuid>
                 │        ├─ eventmask <ncuid>
                 │        ├─ distribution <ncuid>
                 │        ├─ history <ncuid>
                 │        ├─ livelist <ncuid>
                 │        └─ vme <ncuid>
                 ├─ ptk ──┬─ counters <ptkid>
                 │        ├─ readtable <ptkid>
                 │        ├─ updtable <ptkid>
                 │        ├─ creadtable <start_index> <count> <cis_pid>
                 │        ├─ cupdtable <start_index> <count> <cis_pid>
                 │        └─ cstatus <connection_id> <cis_pid>
                 └─ rsu ──┬─ names <rsuid>
                          ├─ load <rsuid> <time>
                          ├─ status <rsuid> <channel>
                          ├─ rx−trace <rsu_id> <channel> <sap> <timeout>
                          └─ tx−trace <rsu_id> <channel> <sap> <timeout>

set ──┬── date ─┬─ <dd.mm.yy> ──┬── <weekday>─── (<hh:mm:ss>)
      │         ├─ <mm/dd/yy> ──┤
      │         └─ <yy−mm−dd> ──┘
      ├── time <hh:mm:ss>
      └── from ─┬─ dcu ────┬─ startservicetime <dcuid>
                │          └─ endservicetime <dcuid>
                ├─ fbc ────┬─ startservicetime <fbcid>
                │          └─ endservicetime <fbcid>
                ├─ kernel ─┬─ mask <cpuid> <value>
                │          └─ globalmask <value>
                ├─ ncu ────┬─ mask <ncuid> <value>
                │          └─ globalmask <value>
                └─ ptk ────┬─ startdebug <ptkid>
                           ├─ enddebug <ptkid>
                           ├─ errorlogdevice <ptkid> <printername>
                           └─ cwrite <index> <cis_pid> <data>
togglemode
systemmode
applmode
disconnect
lock
unlock
user ────┬─ add <userid> <password> <usertype> <codeversion>
         ├─ modify password <userid> <password>
         └─ remove <userid>

capture ─┬─ <filename>
         └─ off

script ──┬─ <filename>
         └─ off

sleep ───── <delay [ms]>
```

5.4 Description of System Mode Commands

5.4.1 clear

clear eventhistory

Clears the Alarms and Events Server history buffer and resets the overflow counter.

5.4.2 clear from dcu

clear from dcu addr <dcuid>

At power−up the DCU listens the Damatic Classic control room bus to get the bus number. If the DCU has been connected to the wrong bus, this command allows you to reset the bus address and make the DCU to listen to a new bus number.

clear from dcu counters <dcuid>

Resets the error counters of the DCU specified by the parameter <dcuid>.

clear from dcu servicetime <dcuid>

Resets the service time counter at the DCU specified by the parameter <dcuid>.

clear from dcu status <dcuid>

Resets the status data of the DCU specified by the parameter <dcuid> and causes the DCU to update the data.

clear from dcu traffic <dcuid>

Resets the Damatic Classic traffic counters of the DCU specified by the parameter <dcuid>.

5.4.3 clear from fbc

clear from fbc counters <fbcid>

Resets the counters of the FBC specified by <fbcid> and empties the history buffer.

clear from fbc io−diag <fbcid> <pic>

Resets the diagnostic data in the 3−page of I/O units in the interface bus of the PIC specified by parameter <pic> by writing a reset command on the 2−page.

clear from fbc pic−counters <fbcid> <pic>

Resets the diagnostic data of the PIC specified by parameter <pic>.

clear from fbc rupi−counters <fbcid>

Resets the RUPI circuit counters in dual port memory of the FBC specified by the parameter <fbcid>.

clear from fbc servicetime <fbcid>

Resets the FBC service time count specified by parameter.

5.4.4 clear from ncu

clear from ncu eventhistory <ncuid>

Clears the event buffer of the NCU specified by parameter <ncuid>.

5.4.5 clear from rsu

clear from rsu counters <rsuid>

Resets the error and message counters of RSU 6 specified by parameter <rsuid>.

5.4.6 kill

kill <procid>

Kills the process specified by the parameter <procid>.

5.4.7 logout

logout

Resets the Diagnostics Server. Do not use this command for exiting the Debugger tool; use the disconnect command instead.

5.4.8 modify memory cpu

modify memory cpu byte <addr(hex)> (<cpuid>)

Substitutes the given 8−bit bytes for a range of bytes in the default CPU or the CPU specified by parameter <cpuid> starting from the address <addr(hex)>.

modify memory cpu word <addr(hex)> (<cpuid>)

Substitutes the given 16−bit words for a range of words in the default CPU or the CPU specified by <cpuid> starting from the address <addr(hex)>.

modify memory cpu long <addr(hex)> (<cpuid>)

Substitutes the given 32−bit long words for a range of long words in the default CPU or the CPU specified by <cpuid> starting from the address <addr(hex)>.

5.4.9 modify memory

modify memory io <fbcid> <pic> <unit> <page> <addr(hex)> <count(hex)> <value(hex)>

Modifies the memory of I/O units according to the given parameters.

Parameter <unit> is the unit number, <page> page number, <addr(hex)> address in the page, <count(hex)> number of channels to be modified and <value(hex)> the new value to be set.

For each modified channel a return value is displayed:

1 = modification was successful
0 = modification failed

5.4.10 modify memory pic

modify memory pic external <fbcid> <pic> <addr(hex)> (<value(hex)>) (<value(hex)>) ...

Modifies the PIC’s RUPI processor external RAM starting from the address <addr(hex)>.

modify memory pic internal <fbcid> <pic> <addr(hex)> (<value(hex)>) (<value(hex)>) ...

Modifies the PIC’s RUPI processor nternal RAM starting from the address <addr(hex)>.

5.4.11 modify line

modify line baudrate <value> <linename>

Sets the baud rate of the serial communications line <linename> to the rate specified by the parameter <value>.

modify line flowcontrol <value> <linename>

Sets or resets XON−XOFF handshake protocol for the serial communications line <linename>.

modify line mode <value> <linename>

Sets the mode of the serial communication line <linename> to the mode defined by the parameter <value>. With the mode parameter you can change the number of data bits and stop bits in the serial line and specify parity check or no parity check and the parity used (’Odd’ or ’Even’).

5.4.12 modify pic

modify pic counters <fbcid> <pic> <index> (<value>) (<value>) ...

Modifies PIC counters (256 counters, numbered 0 − 255) starting from the counter specified by the parameter <index>.

modify pic global <fbcid> <pic> <index> (<value(hex)>) (<value(hex)>) ...

Modifies PIC global variables starting from the variable specified by the parameter <index>.

modify pic state <fbcid> <pic> <index> (<value>) (<value>) ...

Modifies PIC:n status variables starting from the variable specified by the parameter <index>.

5.4.13 print

print background

Prints all currently executed commands.

print directory name <path> (<cpuid>)

Prints the contents of the specified name server directory.

print eventhistory

Prints the Alarms and Events Server history buffer and the value of the buffer overflow counter.

The history buffer can hold 20 event messages. If the number of event messages exceeds 20, the first 10 messages will be preserved and the contents of the remaining 10 locations will be scrolled. The overflow counter indicates the number of lost event messages.

The history buffer can be cleared and the overflow counter reset by the command ’clear eventhistory’.

print grouplist (<groupid>)

Prints a list of group members specified by the parameter <groupid>.

NOTE!
The command will not function in the DIA−PC node.

print line <linename>

Prints the parameters of the serial communication line specified by <linename>.

print memory <addr(hex)> (<cpuid>)

Prints the contents of address <addr> in the CPU specified by the parameter <cpuid> or in the default CPU.

print structure <ncuid>

Prints the hardware configuration of the bus segment specified by the parameter.

print time

Prints the date and time received from the DIA node’s local NCU.

```
print time
Fri May 26 06:52:25.764 1997 v−64
```

print recursive ncu_time <ncuid> <n>

Lists the time from the bus.

The first parameter <ncuid> is the NCU from which is started.
The second parameter <n> tells how many bridges has to crossed (default value 0 − only one's own bus)

The fields to be printed:

- NCU hw address (1)
- NCU software version (2)
- the time in the counter of clock circuit
- the time in the RAM of the clock circuit

```
NCU time form root 0xd1640000
(ncuId:version valid-flags-Mlevel-ownMlevel counters,ram)

--- bus from 0xd164 ---
d164:506 1-00-00-00 C17-05-26 06:58:48.529 5Vc0, R17-05-26 06:58:48.528 5Vc0
d964:506 1-00-00-01 C17-05-26 06:58:48.599 5Vc0, R17-05-26 06:58:48.597 5Vc0
d1a6:506 1-00-00-01 C17-05-26 06:58:48.679 5Vc0, R17-05-26 06:58:48.677 5Vc0
d1c5:506 1-00-00-01 C17-05-26 06:58:48.779 5Vc0, R17-05-26 06:58:48.774 5Vc0
d226:506 1-00-00-01 C17-05-26 06:58:48.859 5Vc0, R17-05-26 06:58:48.858 5Vc0
d125:506 1-00-00-01 C17-05-26 06:58:48.949 5Vc0, R17-05-26 06:58:48.945 5Vc0
d086:506 1-00-00-01 C17-05-26 06:58:49.089 5Vc0, R17-05-26 06:58:49.083 5Vc0
d406:506 1-00-00-01 C17-05-26 06:58:49.209 5Vc0, R17-05-26 06:58:49.201 5Vc0
d426:506 1-00-00-01 C17-05-26 06:58:49.319 5Vc0, R17-05-26 06:58:49.315 5Vc0
d0a6:506 1-00-00-01 C17-05-26 06:58:49.449 5Vc0, R17-05-26 06:58:49.444 5Vc0
e526:506 1-00-00-01 C17-05-26 06:58:49.539 5Vc0, R17-05-26 06:58:49.537 5Vc0
d5c6:506 1-00-00-01 C17-05-26 06:58:49.639 5Vc0, R17-05-26 06:58:49.634 5Vc0
d643:506 1-00-00-01 C17-05-26 06:58:49.759 5Vc0, R17-05-26 06:58:49.758 5Vc0
--- bus from 0xd644 ---
d644:506 1-00-01-01 C17-05-26 06:58:50.129 5Vc0, R17-05-26 06:58:50.129 5Vc0
e164:506 1-00-01-02 C17-05-26 06:58:50.269 5Vc0, R17-05-26 06:58:50.269 5Vc0
e504:506 1-00-01-02 C17-05-26 06:58:50.449 5Vc0, R17-05-26 06:58:50.442 5Vc0

d1c5:506 1-00-00-01 C17-05-26 06:58:48.779 5Vc0, R17-05-26 06:58:48.774 5Vc0
    1        2                  3                            4
```

5.4.14 print from cpu

print from cpu load (<cpuid>)

Prints the loading of the CPU indicated by the parameter or the own CPU.

The loading is given in percents for three time intervals: 1 second, 10 seconds and 1 minute. For each loading value a time label is shown, indicating the time when the value was calculated and stored.

print from cpu machineinfo (<cpuid>)

Prints the type data of the CPU specified by the parameter or the default CPU.

print from cpu peripherals (<cpuid>)

Prints the peripherals data of the CPU specified by the parameter or the default CPU.

print from cpu vme (<cpuid>)

Prints the VME configuration visible to the CPU specified by the parameter or the default CPU.

print from cpu time (<cpuid>)

Prints the date and time of the CPU specified by the parameter <cpuid>.

```
print from cpu time 1610000
Fri May 26 06:52:25.764 1997 v−64
```

print from cpu network delay (<target cpuid> <source cpuid>)

Prints the time a message takes to travel between CPUs given as parameters.

```
print from cpu time 1610000
Fri May 26 06:52:25.764 1997 v−64
```

NOTE!
The command will not function in the DIA−PC node.

5.4.15 print from dcu

print from dcu allgroups <dcuid>

Prints the measurement and output groups loaded in the DCU specified by the parameter.

print from dcu blockmessage <dcuid> <bus(hex 1−F)> <station(hex 1−E)> <message(0−3 or a)>

Prints the contents of the block message specified by the parameter in the DCU block message table. The signal groups to be displayed can be defined 0...3 or ’a’ for displaying all groups.

print from dcu classicbus <dcuid>

Prints the Damatic Classic bus status data, for example, which buses are active, which buses are standby and whether the DCU is connected to the Damatic Classic system.

print from dcu errors <dcuid>

Prints the error counters of the DCU specified by the parameter.

print from dcu group <dcuid> <group>

Prints data on measurement and output groups and the members of these groups.

print from dcu history <dcuid>

Prints the event history of the calculation software in the DCU specified by the parameter.

print from dcu io−modules <dcuid>

Prints the I/O modules loaded in the DCU specified by the parameter.

print from dcu load <dcuid>

Prints the processor loading in the DCU specified by the parameter.

print from dcu moduledata <dcuid> <addr(hex)>

Prints data on the specified I/O module channels and ports using the hw address.

print from dcu servicetime <dcuid>

Prints the response time measurements of the calculation software in the DCU specified by the parameter.

print from dcu stations <dcuid>

Prints the active Damatic Classic automation system process stations connected to the DCU specified by the parameter.

print from dcu version <dcuid>

Prints the module versions of the DCU specified by the parameter.

print from dcu traffic <dcuid>

Prints the Damatic Classic traffic counters data of the DCU specified by the parameter <dcuid>:

- the counter of block messages received
- the counter of block messages read
- the counter of single messages received
- the counter of single messages transmitted
- the counter of single messages passed

5.4.16 print from fbc

print from fbc load <fbcid>

Prints the processor loading in the FBC specified by parameter <fbcid>.

print from fbc rupi counters <fbcid>

Prints the RUPI circuit error counters in the dual port memory of the FBC specified by parameter <fbcid>.

5.4.17 print from fbc calculation

print from fbc calculation allgroups <fbcid>

Prints the measurement and output groups loaded in the FBC specified by parameter <fbcid>.

print from fbc calculation counters <fbcid>

Prints the error counters of the calculation software in the FBC specified by parameter <fbcid>.

print from fbc calculation fbcstates <fbcid>

Prints the states and parameters related with redundancies of the specified FBC.

print from fbc calculation group <fbcid> <groupid>

Prints data on measurement and output group.

print from fbc calculation history <fbcid>

Prints the event history of the calculation software in the FBC specified by parameter <fbcid>.

print from fbc calculation io−module <fbcid>

Prints the I/O modules loaded in the FBC specified by parameter <fbcid>.

print from fbc calculation rolehistory <fbcid>

Prints the last 20 role changes for the specified FBC.

print from fbc calculation servicetime <fbcid>

Prints the response time measurements of the calculation software in the FBC specified by parameter <fbcid>.

5.4.18 print from fbc io

print from fbc io diagnostic <fbcid> <pic>

Prints the 3−page diagnostic data of the I/O units connected to field bus PIC <pic> of the FBC specified by parameter <fbcid>.

print from fbc io memory <fbcid> <pic> <unit> <page> <count>

Prints I/O unit memory of one I/O unit. Parameter <count> specifies the number of channels to be printed. If the unit is not found or the specified number of channels is too large, instead of the data for these channels * characters will be printed. <count> can range 1 − 64.

print from fbc io page <fbcid> <pic> <page> <count>

Prints a number of channels indicated by <count> in the memory page <page> of all I/O units connected to the PIC <pic>.

print from fbc io structure <fbcid>

Prints the field bus configuration in the field bus controller FBC specified by parameter <fbcid>.

print from fbc io version <fbcid>

Prints all I/O unit versions.

5.4.19 print from fbc traffic

print from fbc traffic counters <fbcid>

Prints the field bus counters of the bus connected to FBC specified by parameter <fbcid>.

print from fbc traffic history <fbcid>

Prints the last 20 events from the error counters of the field bus connected with the FBC defined by <fbcid>.

print from fbc traffic load <fbcid> <timeinterval>

This command reads the transmit counters of the FBC specified by parameter <fbcid> twice during the interval <timeinterval>. Then the traffic during the measuring period and the overall situation are printed (request blocks, signals and frames). The time interval is given in seconds.

5.4.20 print from pic

print from pic counters <fbcid> <pic> (<index> <count>)

Prints a number of PIC counters as specified by parameter <count> starting from the counter specified by parameter <index>. The counters are numbered 1 − 256, thus the printing defaults to 16 first counters.

print from pic global <fbcid> <pic> (<index> <count>)

Prints a number of PIC global variables as specified by <count> starting from the variable specified by parameter <index> and the PIC version number. Printing defaults to 16 first variables.

print from pic state <fbcid> <pic> (<index> <count>)

Prints PIC status variables. Presently there is only one variable: IPUFAIL.

print from pic version <fbcid>

Prints the versions of all PICs on the field bus.

5.4.21 print from fbc pic memory

print from fbc pic memory external <fbcid> <pic> <addr(hex)> (<count(hex)>)

Prints a number of PIC's external RAM memory locations as specified by <count(hex)> starting from the address <addr(hex)>.

print from fbc pic memory internal <fbcid> <pic> <addr(hex)> (<count(hex)>)

Prints a number of PIC's internal RAM memory locations as specified by <count(hex)> starting from the address <addr(hex)>.

print from fbc pic memory rom <fbcid> <pic> <addr(hex)> (<count(hex)>)

Prints a number of PIC's ROM memory locations as specified by <count(hex)> starting from the address <addr(hex)>.

5.4.22 print from kernel

print from kernel counters (<cpuid>)

Prints operating system counters from the processor unit specified by <cpuid> or from the default CPU.

print from kernel descriptor <procid>

Prints a process descriptor specified by parameter <procid>.

print from kernel eventmask (<cpuid>)

Prints the event message mask from the processor unit specified by parameter <cpuid> or from the default CPU.

print from kernel groupidlist (<cpuid>)

Prints the list of reserved group identifiers at the processor unit specified by parameter <cpuid> or at the default CPU.

print from kernel information (<cpuid>)

Prints some operating system constants from the processor unit specified by parameter <cpuid> or from the default CPU.

print from kernel logicalidlist (<cpuid>)

Prints the list of reserved logic identifiers in the processor unit specified by parameter <cpuid> in the default CPU.

print from kernel memory (<cpuid>)

Prints operating system memory parameters from the processor unit specified by parameter <cpuid> or from the default CPU.

print from kernel processlist (<cpuid>)

Prints the process list from the processor unit specified by parameter <cpuid> or from the default CPU.

print from kernel status (<cpuid>)

Prints variables indicating the state of the operating system from processor unit specified by parameter <cpuid> or from the default CPU.

print from kernel teamstatus <procid>

Prints the data on a given team in the process specified by parameter <procid>.

5.4.23 print from ncu

print from ncu counters <ncuid>

Prints the counters of the NCU specified by parameter <ncuid>.

print from ncu eventmask <ncuid>

Prints the event mask of the NCU specified by parameter <ncuid>.

print from ncu distribution <ncuid>

Prints the message length distribution at the NCU specified by parameter <ncuid>.

print from ncu history <ncuid>

Prints the event history of the NCU specified by parameter <ncuid>.

print from ncu livelist <ncuid>

Prints the ’LiveList’ table of the NCU specified by parameter <ncuid>. This table contains the paired addresses of the bus controllers (frame + VME address) on the respective process bus and the ’token’ addresses.

print from ncu vme <ncuid>

Prints the VME configuration visible to the NCU specified by parameter <ncuid>.

5.4.24 print from ptk

print from ptk counters <gtwid>

Prints the event counters related with the message traffic (process computer < − > GTW:CIS) of a Computer Interface Server (CIS) using the TEK protocol. The <gtwid> parameter specifies the Computer Interface Server whose data you want to monitor (e.g. a10000).

print from ptk readtable <gtwid>

Prints all configured connections from the read table of a Computer Interface Server (CIS) using the TEK protocol. Indexing starts at 0. The <gtwid> parameter specifies the Computer Interface Server (e.g. a10000).

The index, Valmet DNA address in decimal and hexadecimal format, and the data in decimal and hexadecimal format are printed from the table on consecutive lines.

print from ptk updtable <gtwid>

Prints all configured connections from the update table of a Computer Interface Server (CIS) using the TEK protocol. Indexing starts at 0. The <gtwid> parameter specifies the Computer Interface Server (e.g. a10000).

The index, Valmet DNA address in decimal and hexadecimal format, and the data in decimal and hexadecimal format are printed from the table on consecutive lines.

print from ptk creadtable <start_index> <count> <cis_pid>

This command prints values from the read table of a Computer Interface Server (CIS) using the CNP protocol. Printing starts at the address defined by <start_index>, and <count> defines the number of values to be printed. The <cis_pid> parameter specifies the Computer Interface Server (e.g. a10000).

The command prints the data type (e.g. ana or bin) and the variable’s value.

print from ptk cupdtable <start_index> <count> <cis_pid>

This command prints values from the update table of a Computer Interface Server (CIS) using the CNP protocol. Printing starts at the address defined by <start_index>, and <count> defines the number of values to be printed. The <cis_pid> parameter specifies the Computer Interface Server (e.g. a10000).

The command prints the data type (e.g. ana or bin) and the variable’s value.

print from ptk cstatus <connection_id> <cis_pid>

This command prints counter values for CNP protocol’s message traffic. The <cis_pid> parameter specifies the desired Computer Interface Server (CIS). The <connection_id> parameter specifies the monitored channel as follows:

0 or >= (N + 1)
Print the summed values of all channels.

1...N
Print the counters of the specified channel (also RSU channel number, sap number, RSU pid, and CPU pid that serves the specified channel; in connection with Ethernet, this will print the pid of the process on Ethernet unit, the CPU pid serving the specified channel, and the number of the TCP/IP port).

5.4.25 print from rsu

print from rsu names <rsuid>

This command prints the names that exist on the unit and are loadable: protocols, drivers and crc functions. A name existing in the list must be selected always when initializing a channel.

print from rsu load <rsuid> <time>

This command prints the traffic load of all six channels during the interval defined by <time>. The execution of the command always takes the time defined by <time> in seconds. The values for a single channel’s counters are printed on one line: channel number, messages sent, number of bytes in the messages (effective bytes), messages received, number of bytes in received messages, number of messages sent and received, and the number of bytes sent and received (in the last field).

If you define 10 seconds as <time>, the value of the last column will correspond to the channel’s effective baud rate (bit/s) fairly closely.

The characters used for framing the messages are not included in these values. The last field of the Total line shows the summed load of all channels. In the current version the maximum is approx. 4*19200 = 76800, after which the unit may lose characters when receiving messages.

The loading of RSU6 unit’s processor can be read in the same way as from CPU: ’print from cpu load <rsuid>’.

print from rsu status <rsuid> <channel>

This command prints the parameters and errors for a single channel of RSU6.

The version number of the program residing on the unit is printed first.

General−purpose counters indicate the number of messages sent through the channel after start−up or after resetting the counters.

The counters indicating the memory and buffer status show the unit's memory capacity in kilobytes, the amount of reserved memory space, and instantaneous value. In addition, they show the number of channel−specific buffers.

General error counters are printed next (if there are any errors).

The success of the configuration can be checked from the protocol's parameters. Possible errors are printed after the parameter data. The driver parameters and the values of error counters are then printed in the same way (i.e., if counter value = 0, it will not be printed).

The data of the LSAPs (Link level Service Access Points) opened for the channel is printed last. Several SAPs having a different owner process cannot be opened for a single channel.

print from rsu rx−trace <rsuid> <channel> <sap> <timeout>

This command takes one message as a sample and displays the message in hexadecimal and ascii format.

The channel number (1...6), SAP's number (0...63) and timeout in seconds (1...655) are given as command parameters. The last two values can be omitted, in which case the software will trigger on any received message using the maximum time of 10 seconds. If the message is received within the required time, the message sender's ssap (source sap), the receiver's dsap (destination sap) and message length will be printed first. The message's data field is then printed in such a way that the data field's byte counter is at the left edge in hexadecimal format.

The program only displays successfully received messages. Message transfer to CPU must also be successful.

print from rsu tx−trace <rsuid> <channel> <sap> <timeout>

The same as above, but this command involves transmitted messages.

5.4.26 set

set date <dd.mm.yy>/<mm/dd/yy>/<yy−mm−dd> <weekday> (<HH:MM:SS>)

Sets the date and time (if specified) in the real time clock according to the parameter.

There are three alternative syntaxes for the date: ”dd.mm.yy” or ”mm/dd/yy” or ”yy−mm−dd”.

The date and time are tested to ensure that their values are within the valid range:

1 <= dd <= 31
1 <= mm <= 12
0 <= yy <= 99
0 <= HH <= 23
0 <= MM <= 59
0 <= SS <= 59

<weekday> is from 1 to 7, where 1=monday and 7=sunday.

set time <HH:MM:SS>

Sets the time in the real time clock according to the parameter. The values are checked as in the previous command.

5.4.27 set from dcu

set from dcu startservicetime <dcuid>

Starts the DCU service time counter specified by the parameter.

set from dcu endservicetime <dcuid>

Stops the DCU service time counter specified by the parameter.

5.4.28 set from fbc

set from fbc startservicetime <fbcid>

Starts the FBC service time counter specified by the parameter.

set from fbc endservicetime <fbcid>

Stops the FBC service time counter specified by the parameter.

5.4.29 set from kernel

set from kernel mask <cpuid> <value(hex)>

Sets the operating system event mask in the CPU specified by the parameter <cpuid> according to the given parameter.

The event mask determines what events cause an event message to be sent to the Alarms and Events Server.

On power−up the events mask is set to prevent the sending of event messages.

The mask bits stand for:

bit 0 = timeout
bit 1 = nonExistProc
bit 2 = powerFail

The parameter is given as a hexadecimal number with one or several mask bits set.

set from kernel globalmask <value(hex)>

Sets the operating system event masks of all CPUs in Valmet DNA according to the parameter. Operation as above.

5.4.30 set from ncu

set from ncu mask <ncuid> <value(hex)>

Sets the event mask of the NCU specified by the parameter <ncuid> according to the given parameter.

The event mask determines what events cause an event message to be sent to the Alarms and Events Server.

On power−up the events mask is set to prevent the sending of event messages.

The mask bits stand for:

bit 0 = buffer
bit 1 = timeout
bit 2 = noise
bit 3 = hdrAbort
bit 4 = hdrOverrun
bit 5 = hdrFrame
bit 6 = hdrCRC
bit 7 = tooShort
bit 8 = sender
bit 9 = frmAbort
bit 10 = frmOverrun
bit 11 = frmFrame
bit 12 = frmCRC
bit 13 = lengthErr
bit 14 = tooLong
bit 17 = LLAppear
bit 18 = LLDisappear

The parameter is given as a hexadecimal number with one or several mask bits set.

set from ncu globalmask <value(hex)> (<ncuid>)

Sets the event masks of all NCUs in Valmet DNA according to the parameter. Operation as above.

5.4.31 set from ptk

set from ptk startdebug <gtwid>

Starts a test print to the Application Server console line at the Computer Interface Server (CIS) specified by the parameter <gtwid>.

set from ptk enddebug <gtwid>

Ends the test print at the Computer Interface Server (CIS) specified by the parameter <gtwid>.

set from ptk errorlogdevice <gtwid> <printername>

Sets the error log device for the Computer Interface Server (CIS) specified by the parameter <gtwid> according to the parameter <printername>.

set from ptk cwrite <index> <cis_pid> <data>

This command writes the value <data> at the position defined by <index> in the update table of a Computer Interface Server (CIS) using the CNP protocol. The <cis_pid> parameter specifies the Computer Interface Server (e.g. a10000). The success of the write operation can be checked with the command ’print from ptk updtable’ (you should then remember that the computer may already have changed the value in question).

5.4.32 togglemode

togglemode

Toggles between system mode and application mode.

5.4.33 systemmode

Systemmode commands switches Debugger to system mode.

5.4.34 applmode

Applmode commands switches Debugger to application mode.

5.4.35 disconnect

disconnect

Exits the Debugger tool.

5.4.36 lock

lock

Locks a session preventing it to be aborted by another user.

5.4.37 unlock

unlock

Unlocks a session.

5.4.38 user

user add <userid> <password> <usertype> <codeversion>

Inserting an user identification and assigning it a password, user type and code version.

user remove <userid>

Removes the user identifier specified by the parameter.

5.4.39 user modify

user modify password <userid> <password>

Changes the password of a user identifier.

5.4.40 capture

capture <filename>

The capture command stores the user's commands and their printouts into a file. Storing is activated by giving command 'capture <filename>' and deactivated by command 'capture off'. The storing file can be directed to a local hard disk (debugger in a PC node) or a Backup Server.

Examples:

```
capture c:\temp\my_capture.txt
```

```
capture :s:AB01:my_capture.txt
```

If the file name does not include the path, for example "c:\temp", the storing file is created in the same place where the Debugger tool has been launched from (in a PC node the default is D:\dna\ca\dia).

capture off

Capture off commad deactivates storing the printouts into the file and cloces storing file. Stored file can be viewed only when file has been closed with the capture off command.

5.4.41 script

script <filename>

The script command reads command from a file and executes them. Script command is given in the form of 'script <filename>' where the 'filename' is the file's name which contains the commands. The executed commands can be any kind normal Debugger commands.

An example of a file run with the script command, where:

- printing mode is set as 'compact'
- the commands and printouts are directed to bu server's file 'TX_out.txt'
- the diagnostics sensor member is listed twice with a 10 second delay in between

```
set printsty compact
capture :s:AB01:TX_out.txt
applmode
p v :e:di:AP01:node06:chan1:tx_count
sleep 10000
p v :e:di:AP01:node06:chan1:tx_count
cap off
```

5.4.42 sleep

sleep <delay [ms]>

The sleep command causes a delay, during which the Debugger does not handle commands. Sleep is used to create delays between commands in command scripts, which are run with the script command. The length of the delay is given in milliseconds, for example the command

```
sleep 10000
```

causes a 10 second delay.

6 Diagnostics Sensors

6.1 General

Diagnostics sensors are data updated by Valmet DNA, which contain information on the internal Valmet DNA quantities such as the extent of application configuration, etc. The sensors are named and thus also typed. The application engineer does not need to configure the sensors, Valmet DNA creates them automatically on the basis of the Application Server type.

The diagnostics sensors have primarily been designed for the following functions:

- to serve Valmet DNA maintenance (e.g. debugging)
- to serve application modification (especially by showing the resources that limit the modification, thus allowing you to determine what chances of success the modification will have)
- to provide a connection from applications to fault information, permitting application−specific response to different failures

Valmet DNA system alarms are formed using the diagnostics sensors. Control room monitors receive data on the following events:

- connection to node is lost
- power supply failure in the node and node switches to battery supply
- Application Server starts up and communicates with the other components
- PCS node loses connection with I/O rack
- I/O unit is faulty: no connection with the unit can be made or the I/O unit has detected an error in its own process interface
- main/reserve controller switchovers at redundantly configured controllers, and problems in updating the passive controller

The data contained in the diagnostics sensors can be examined with the Diagnostics Server Debugger tool or they can be connected to the application. The sensors are named data and of the form

```
:e:di:<component name>:<directory>:<data name>
```

therefore their values can be examined with the Debugger tool by giving the following application mode command:

```
5a% print variable :e:di: ...  <CR>
```

In a redundantly configured PCS the sensors are under two names. The name shown above is used to read the sensor at the active controller. The following names are used to read the sensor from main and reserve controller, respectively:

```
:e:main:<component name>:<directory>:<data name>

:e:reserve:<component name>:<directory>:<data name>
```

The following paragraphs present the diagnostic sensors related to different units in more detail.

6.2 Process Bus Description

The process bus structure is described with a sensor whose name is as follows:

```
:e:di:<component name>:slot<I>:vmecard
```

The sensor’s type is ’**dnst**’. This sensor is found in process bus’s each component, and it shows how each component sees the process bus to which it is connected.

The following alarm is received through the sensor:

`<time> MAIN SYSTEM BUS <nn> FAULT`	(member ’sbalive’)
`<time> RES. SYSTEM BUS <nn> FAULT`	(member ’sbalive’)

Data structure for type ’**dnst**’:

type

Type:	uns16
Default:	0
Description:	Type of plug−in unit from which the structure is read:

1 = CPU
2 = FBC
3 = NCU
4 = DCU
5 = reserved
6 = RSU6

version(8)

Type:	char
Default:	’ ’
Description:	Plug−in unit’s software version.

ncucount

Type:	ints
Default:	48 0
Description:	Number of NCUs on component’s process bus.

stations(50)

Type:	stid
Default:	
Description:	The Application Servers on component’s process bus.

’stations(xx):name’ gives the Application Server name; ’stations(xx):conn’ shows whether connection to the Application Server is O.K. (as well as the time at which the connection was last established or lost).

sbalive

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Connection status to process bus.

’sbalive:binstat’:
0 = no connection
1 = connection O.K.

msgcount

Type:	intl
Default:	48 0
Description:	Number of messages communicated after NCU’s start−up.

errcount

Type:	intl
Default:	48 0
Description:	Number of transmission errors after NCU’s start−up.

loadinc

Type:	ints
Default:	48 0
Description:	NCU’s load effect on the bus.

noise

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Transmission noise.

’noise:binstat’:
0 = no noise
1 = transmission noise

Application Server’s visibility on process bus is described with a type stid.

Type ’**stid**’ data structure:

name(10)

Type:	char
Default:	
Description:	Application Server’s name

conn

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Connection status to Application Server

Data ’conn:binstat’:
0 = no connection to Application Server
1 = connected to Application Server

6.3 Channel Status

When buses are connected with the help of RSU6 additional information of process bus status is available with a sensor whose name is:

`:e:di:<component name>:slot<i>:chan<j>` where j = 1...6

The sensor type is ’drnst’. It shows the status of one RSU6 serial channel.

The data structure for type **’drnst’**:

ch

Type:	ints
Default:	48 0
Description:	Channel index

speed

Type:	intl
Default:	48 0
Description:	Channel speed

target(16)

Type:	char
Default:	
Description:	Target unit

chalive

Type:	binev
Default:	
Description:	Connection status

Data chalive:binstat
0 = no connection to target
1 = connected to target

msgcount

Type:	intl
Default:	48 0
Description:	Number of message transactions since startup

errcount

Type:	intl
Default:	48 0
Description:	Number of transmission failures since startup

loadinc

Type:	ints
Default:	48 0
Description:	Bus load caused by unit (%)

noise

Type:	binev
Default:	
Description:	Transmission problems

Data noise:binstat
0 = no noise
1 = transmission problems

6.4 Application Server Description

The name of the sensor representing a VME rack and its Application Server is

```
:e:di:<application server name>:activity
```

and its type is **’dacty’**.

The sensor contains general data on the Application Server and a description of the configuration of the rack where the Application Server is located.

Through this sensor we get the following system alarms:

```
<time> <nn> STATION DOES NOT ANSWER        (member ’fault’)

<time> <nn> STATION IN STANDBY STATE       (member ’stdby’)

<time> <nn> STATION HAS STARTED            (member ’startup’)
```

Type **’dacty’** data structure:

type

Type:	uns16
Default:	0
Description:	Indicates Application Server type

0 = unidentified Application Server
1 = Process Control Server
2 = Operation Server
3 = Alarms and Events Server
4 = Trend Server
5 = Recipe Server
6 = Damatic Interface Server
7 = Computer Interface Server
8 = Router Server
9 = Diagnostics Server
10 = Backup Server
11 = Logic Interface Server
12 = Quality Control Server
13 = generic communication server
14 = Netwatch
15 = Web Diagnostics tool
16 = free protocol server
17 = systerm mode querier

startup

Type:	binev
Default:	49 0 0 0 0 0 1 1 1 00 14
Description:	Indicates the Application Server startup time

In addition the ’startup:binstat’ data allows examining the Application Server run data
0 = Application Server running
1 = Application Server starting

stdby

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	

Indicates whether the VME rack in which the Application Server is located gets its operating voltage from the AC supply or from the batteries. Also contains the time when supply switched to batteries or back to mains.

Values of ’stdby:binstat’:
0 = AC supply
1 = supply by batteries

units

Type:	uns16
Default:	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Description:	Contains a list of all plug−in units in the VME rack

0 = no plug−in unit visible to VME
1 = CPU
2 = FBC
3 = NCU
4 = DCU
5 =SBC
6 = RSU6
7 = GDC
8 = memory size 1 Mbytes
9 = memory size 2 Mbytes
10 = memory size 4 Mbytes
11 = memory size 8 Mbytes
99 = unidentified plug−in unit

fault

Type:	binev
Default:	48 0 0 0 0 0 0 1 1 1 00 14
Description:	Indicates an active fault in the Application Server.

Data ’fault:binstat’:
0 = no fault
1 = active fault

faultind

Type:	intsev
Default:	48 0 0 0 0 0 0 1 1 1 00 14
Description:	Fault data specifier

Data ’faultind:intssta’
0 = no fault
1 = No fault other than a peripheral error preventing the execution of an operation (There is a fault in a device used by the Application Server. As this is a device error, a ’fault’ alarm on this Application Server area will not be generated − the situation is indicated as a consequent event instead.)
2 = operation malfunction (for example trying to read nonexistent file)
3 = buffer overflow
4 = capacity exceeded
5 = insufficient configuration
6 = configuration error in active configuration which can lead to partially improper function of Application Server
7 = cannot load data backup
8 = cannot load configuration backup
9 = system fault
10 = control task execution problem
11 = controller was reset by the PMM
12 = license is about to expire / has expired

6.5 Redundancy Nugget

Each redundantly configured Application Server is provided with a diagnostics nugget (redundancy nugget) that allows you to define the main/reserve controller switchover characteristics and update rate between the controllers, and to give a manual switchover command. The name of this diagnostics nugget is as follows:

```
:e:sn:<component name>:red_ctr
```

The nugget's type is redctr2.

The diagnostics nugget includes the following members:

k_sb

Type:	ints
Default:	48 0
Description:	Defines the controller switchover response for process bus failure.

This member defines for how many seconds the connection between active controller and process bus can be faulty before the operation is switched over to the controller connected to a process bus whose connection is O.K. You can give 5...30 seconds as the time. If the value is 0, there will be no switchover even if the connection were faulty.

k_io

Type:	ints
Default:	48 0
Description:	Defines the controller switchover response for field bus failure.

When the passive Application Server sees more I/O channels in the value displayed in the sensor that active Application Server then the operation is switched over to the controller connected to a field bus that is in a better condition. You can define 1...32000 I/O connections as the value of this member.

command

Type:	intl
Default:	48 0
Description:	Defines manual switchover.

Controller switchover will occur when this member is given the value 171164.

swctrl

Type:	ints
Default:	48 0
Description:	Class A module update in controlled switchover

Module update of class A modules in controlled switchover:

0 = Normal update
1 = No updating

Modules in update classes B and C are not updated.

daterate

Type:	ints
Default:	48 0
Description:	Update ratio of class A modules

The ratio of sampling rate to execution interval of a control task. You can give 1...100 as the value of this member.

daterateb

Type:	ints
Default:	48 0
Description:	Update ratio of class B modules

The ratio of sampling rate to execution interval of a control task. Class B modules are updated every N execution cycles, where N = daterate * daterateb.

updclass_b

Type:	txt64
Default:	
Description:	Class B modules/module mask

To clear the field, type a letter that doesn’t match with module names, e.g. comma. In controlled switchover class B modules are not updated to reserve controller.

updclass_c

Type:	txt64
Default:	
Description:	Class C modules/module mask

To clear the field, type a letter that doesn’t match with module names, e.g. comma. Class C modules are not updated to reserve controller. During switchover modules belonging to update class C are always initialized.

fbcrate

Type:	ints
Default:	48 0
Description:	Update ratio of I/O modules

ctr_wait

Type:	ints
Default:	48 0
Description:	Maximum time used in waiting a control task to finish during switchover in passivating controller <ms>.

smode

Type:	ints
Default:	48 0
Description:	Initialization method after switchover

1 = Only differing modules in configuration are initialized
2 = If control task has differing module, all modules are initialized
3 = All modules in controller are initialized.

See also member updclass_c

method

Type:	ints
Default:	48 0
Description:	Method for copying data to redundant controller

0 = All data is transferred
1 = Only changed data is transferred.

rbuffer

Type:	intl
Default:	48 0
Description:	Size of update buffer in bytes.

If method = 1, this must be greater than size of biggest module

6.6 Application Server Redundancy Description

The redundancy of an Application Server is described by a sensor whose name is as follows:

```
:e:di:<application server name>:redundancy
```

The sensor's type is 'drdy2'**.**

The following messages are received through this sensor:

```
<time> MAIN/RES MACH. PCS <nn> PASS MACH. NOT UP TO DATE

<time> STATION MAIN/RES MACH. <nn> HAS STARTED
```

The data structure for type 'drdy2' is as follows:

position

Type:	bin
Default:	
Description:	Controller's physical position.

0 = main
1 = reserve

role

Type:	ints
Default:	
Description:	This controller's role.

0 = active
1 = deactivating
2 = passive
3 = activating

swover

Type:	intsev
Default:	
Description:	Gives the time and cause for switching over to the current role.

Data 'swover:intsstat:s':
0 = redundant controller faulty at start−up
1 = field bus faulty at start−up
2 = redundant controller already active at start−up
3 = update bus fault
4 = controller as reserve controller at start−up
5 = controller as main controller at start−up
6 = process bus fault
7 = field bus fault
8 = switchover command
9 = redundant controller failure
10 = process bus fault on redundant controller

weights

Type:	bfcoef
Default:	
Description:	Process and field bus failure tolerance times.

’weights:k_sb’ process bus tolerance time
’weights:k_io’ field bus tolerance time

rstatus

Type:	binev
Default:	
Description:	Reserve controller’s condition and the time at which it went to this status.

The values of ’rstatus:binstat’:
0 = redundant controller faulty
1 = redundant controller O.K.

sbalive

Type:	binev
Default:	
Description:	

Controller’s connection to its own process bus and the time at which the connection last returned or was lost.

The values of ’sbalive:binstat’:
0 = process bus connection off
1 = process bus connection O.K.

redsbalive

Type:	binev
Default:	
Description:	

Redundant controller’s connection to its own process bus and the time at which the connection last returned or was lost.

The values of ’redsbalive:binstat’:
0 = process bus connection off
1 = process bus connection O.K.

ownfb

Type:	ints
Default:	
Description:	

Parameters describing the condition of controller’s own field buses and the PICs on the buses (number of alive connections to I/O points via own field buses).

redfb

Type:	ints
Default:	
Description:	

Parameters describing the condition of redundant controller’s field buses and the PICs on the buses (number of alive connections to I/O points via redundant controller’s field buses).

dbalive

Type:	binev
Default:	
Description:	

Update bus connection to reserve controller and the time at which the connection last returned or was lost. Values of ’dbalive:binstat’:

0 = update bus connection off
1 = update bus connection O.K.

daterate

Type:	ints
Default:	
Description:	Update ratio of class A modules

The ratio of sampling rate to execution interval of a control task.

datestatus

Type:	bin
Default:	
Description:	

Shows whether the applications on the controllers are identical. If not, switchover is not allowed.

0 = applications are different
1 = applications are identical

diff

Type:	modidf
Default:	
Description:	

The first different module between the controllers: the module’s name and version in own controller’s module library.

datings

Type:	intsev
Default:	
Description:	

The time at which the controller last transmitted/received updating to/from the reserve controller, and the average amount of data (kB/s) transferred between controllers in updates.

jbalive

Type:	binev
Default:	
Description:	Not in use.

picjbf

Type:	intsev
Default:	
Description:	Not in use.

updclass_b

Type:	txt64
Default:	
Description:	Module mask defining class B modules

updclass_c

Type:	txt64
Default:	
Description:	Module mask defining class C modules

redmode

Type:	txt84
Default:	
Description:	Initialization method after switchover

1 = Only differing modules in configuration are initialized
2 = If control task has differing module, all modules of that task are initialized
3 = All modules in Application Server are initialized

fault

Type:	binev
Default:	
Description:	

Shows a possible fault in updating the reserve controller; also gives the time at which the fault occurred or disappeared. Specified by the ’faultind’ member.

The values of ’fault:binstat’:
0 = no fault
1 = active fault

faultind

Type:	intsev
Default:	
Description:	

Specifies a fault related with reserve controller and gives the time at which the fault occurred or disappeared.

The values of ’faultind:intsstat:s’:
0 = no fault
1 = redundant controller not operative
2 = active controller is unable to send updates at the desired intervals
3 = applications are not identical, and more than 5 minutes have elapsed since the last configuration modification to the differing control task or FBC
4 = no update bus connection to redundant controller
5 = system fault
6 = connection to redundant controller is cut (RCOK cable fault)
7 = redundant controller update cycle is too fast
8 = index reserved but not in use
9 = main and reserve have different amount of I/O channels

faulttext

Type:	txt64
Default:	
Description:	Fault description text.

6.7 Computer Interface Server Description

6.7.1 TEK Protocols

A Computer Interface Server (CIS) using the TEK protocol is described with two sensors, depending on which protocol is actually used. The names of these sensors are as follows:

```
:e:di:<component name>:TEK-126:serial
```

```
:e:di:<component name>:TEK-235:serial
```

The type of both sensors is '**dplc**'. The sensors contain information on serial communications between Computer Interface Server and process computer.

The data structure for type '**dplc**':

speed

Type:	intl
Default:	48 0
Description:	Serial line's transmission speed.

protocol(8)

Type:	char
Default:	' '
Description:	The used protocol.

format(8)

Type:	char
Default:	' '
Description:	Serial line's transmission parameters.

rdatamsgs

Type:	intl
Default:	48 0
Description:	Number of messages received.

sdatamsgs

Type:	intl
Default:	48 0
Description:	Number of messages sent.

retrials

Type:	intl
Default:	48 0
Description:	Number of retransmissions.

rejmsgs

Type:	intl
Default:	48 0
Description:	Number of rejected messages.

lastsuccess

Type:	time
Default:	0 0 0 0 0 1 1 1 00 14
Description:	Time of last successful connection.

fault

Type:	binev
Default:	0 0 0 0 0 0 1 1 1 00 14
Description:	Shows a possibly active fault in the Computer Interface Server.

0 = no active fault
1 = active fault exists

The fault is specified by ’faultind’.

faultind

Type:	intsev
Default:	0 0 0 0 0 0 0 1 1 1 00 14
Description:	Specifies an active fault.

The values of ’faultind:intsstat:s’:
0 = no fault
1 = Computer Interface Server indicates a fault at the server itself
2 = the line is alive, but data transfer fails
3 = process computer does not respond

6.7.2 CNP Protocols

There are two CNP protocols as well:

- ISO−1745 Standard
- ISO−2111 Standard

The nodes have a different physical structure. As a result the naming of the sensors is also slightly different. However, the type of the sensors is the same (’**dplc**’, whose data structure is described in 6.7.1 ”TEK protocols”).

ISO−1745

Besides the CPU, a VME node using the ISO−1745 Standard CNP protocol includes an RSU6 unit. Depending on the location of this unit, the node has one of the following sensors:

If the RSU6 is located in the same rack, the sensor’s name is:

```
:e:di:<component name>:CNP:serial:RSUslot<n>:channel<m>
```

where <n> is the card slot in the rack and <m> is the unit’s channel (1...6).

If the RSU6 is located in a different rack, the sensor’s name is:

```
:e:di:<component name>:CNP:serial:RSU<rsupid>:channel<m>
```

where <rsupid> is the unit’s address and <m> is the unit’s channel (1...6).

For instance, if the node’s lid is AC12, RSU6’s pid is 0x4a050000 and the channel is 4, the sensor’s name will be:

```
:e:di:AC12:CNP:serial:RSU4a05:channel4
```

ISO−2111

A VME node using the ISO−2111 Standard CNP protocol only has the CPU, and the sensor’s name is as follows:

```
:e:di:<component name>:CNP:serial:ISO-2111
```

6.8 Logic Interface Server Description

Logic Interface Server is described with a sensor whose name is as follows:

`:e:sn:UL01_main:dplcX` ,where X = 1...9 is a channel number

The sensor's type is '**dplc**'. The sensor contains information on serial communications between Logic Interface Server and programmable logic.

The data structure for type '**dplc**':

speed

Type:	intl
Default:	48 0
Description:	Serial line's transmission speed.

protocol(8)

Type:	char
Default:	' '
Description:	The used protocol.

format(8)

Type:	char
Default:	' '
Description:	Serial line's transmission parameters.

rdatamsgs

Type:	intl
Default:	48 0
Description:	Number of messages received.

sdatamsgs

Type:	intl
Default:	48 0
Description:	Number of messages sent.

retrials

Type:	intl
Default:	48 0
Description:	Number of retransmissions.

rejmsgs

Type:	intl
Default:	48 0
Description:	Number of rejected messages.

lastsuccess

Type:	time
Default:	0 0 0 0 0 1 1 1 00 14
Description:	Time of last successful connection.

fault

Type:	binev
Default:	0 0 0 0 0 0 1 1 1 00 14
Description:	Shows a possibly active fault in the Logic Interface Server.

The values of ’fault:binstat’:
0 = no active fault
1 = active fault exists

The fault is specified by ’faultind’.

faultind

Type:	intsev
Default:	0 0 0 0 0 0 0 1 1 1 00 14
Description:	Specifies an active fault.

The values of ’faultind:intsstat:s’:
0 = no fault
1 = Logic Interface Server indicates a fault at the server itself
2 = the line is alive, but data transfer fails
3 = process computer does not respond

6.9 VME Plug−in Unit Description

Each plug−in unit operating on VME bus and used by a component is described with a sensor whose name is as follows:

```
:e:di:<component name>:slot<i>:vmecard
```

where <i> is the plug−in unit’s number 1−16 (numbered 01,02,03, etc.). The sensor’s type is ’**dvst**’. Each plug−in unit has a single sensor.

The sensors describe a plug−in unit and its operating system, software version, and the resources used by the unit.

The following alarm is received through the sensor:

```
<time> STATION <nn> FAULT IN CARD <i>
```

where <i> = CPU, FBC, NCU or DCU.

Data structure for type ’**dvst**’:

type

Type:	uns16
Default:	0
Description:	Plug−in unit’s type.

1 = CPU
2 = FBC
3 = NCU
4 = DCU
5 = reserved
6 = RSU6

version(8)

Type:	char
Default:	’ ’
Description:	The plug−in unit version required by the software.

kernel(16)

Type:	char
Default:	' '
Description:	The type and version of the operating system on the plug−in unit.

kermem

Type:	intl
Default:	48 0
Description:	Total size of the memory available to the operating system.

freekermem

Type:	intl
Default:	48 0
Description:	Amount of free memory space available to the operating system.

maxkerblock

Type:	intl
Default:	48 0
Description:	Size of operating system's largest free memory block.

t1

Type:	ints
Default:	48 0
Description:	Unit load in percent in a cycle of last 1 second.

t10

Type:	ints
Default:	48 0
Description:	Unit load in percent in a cycle of last 10 seconds.

t60

Type:	ints
Default:	48 0
Description:	Unit load in percent in a cycle of last 60 seconds.

chann

Type:	uns32
Default:	0
Description:	

Shows which serial ports etc. on the plug−in unit are in use, or which free ones can be applied to the desired use. The channels are divided in accordance with the bits as follows:

bit 0 = RS 1 serial line
bit 1 = RS 2
bit 2 = RS 3
bit 3 = RS 4
bit 4 = RS 5
bit 5 = RS 6
bit 6 = B−1 binary I/O
bit 7 = B−2

The value of each bit is coded as follows:
0 = line is free
1 = line is in use

code(16)

Type:	char
Default:	’ ’
Description:	Type and version of software component loaded to plug−in unit.

types(16)

Type:	char
Default:	’ ’
Description:	Name and version of type set loaded to component.

dynmem

Type:	intl
Default:	48 0
Description:	Total size of dynamically allocatable memory available in component.

freemem

Type:	intl
Default:	48 0
Description:	Amount of dynamically allocatable free memory available in component for use in application program.

maxblock

Type:	intl
Default:	48 0
Description:	Size of largest dynamically allocatable free memory block.

blockcnt

Type:	ints
Default:	48 0
Description:	Total amount of dynamically allocatable free memory blocks.

block0cnt

Type:	ints
Default:	48 0
Description:	Number of free memory blocks in the 10−127 byte size category.

block1cnt

Type:	ints
Default:	48 0
Description:	Number of free memory blocks in the 128−1023 byte size category.

block2cnt

Type:	ints
Default:	48 0
Description:	Number of free memory blocks in the 1024−8191 byte size category.

block3cnt

Type:	ints
Default:	48 0
Description:	Number of free memory blocks in the 8192−65535 byte size category.

resources(24)

Type:	intl
Default:	48 0
Description:	Component’s resources.

’resources(13)’ control tasks
’resources(14)’ I/O groups
’resources(15)’ I/O modules
’resources(16)’ field bus’s loading number on FBC
’resources(17)’ number of norm I/Os on FBC
’resources(18−24)’ local resources

freeresources(24)

Type:	intl
Default:	48 0
Description:	Free resources.

’freeresources(13)’ control tasks
’freeresources(14)’ I/O groups
’freeresources(15)’ I/O modules
’resources(16)’ field bus’s free loading number on FBC
’resources(17)’ number of free norm I/Os on FBC
’freeresources(18−24)’ local resources

fault

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Indicates whether there are any problems in plug−in unit’s operation affecting the component’s operation.

The values of ’fault:binstat’:
0 = no fault
1 = active fault

faultind

Type:	intsev
Default:	48 0 0 0 0 0 0 1 1 1 00 14
Description:	Specifies a fault.

Data ’faultind:intsstat:s’
0 = ok
1 = capacity overload
2 = buffer overflow
3 = out of memory
4 = memory fault
5 = system fault
6 = no answer
7 = data transfer fault to external system

6.10 I/O Rack Description

Each I/O rack assigned to the component is described by a sensor whose name is

```
:e:di:<component name>:slot<i>:iorack<j>
```

where <i> is the FBC's number (numbered 01, 02, 03 etc.) and <j> is the I/O rack's electrical address (j = 0...15). The sensor's type is '**dirk**'.

For each FBC there are 16 sensors. The data received from sensors indicate what I/O racks there are on the field bus. The sensors also contain data on the plug−in units located in the I/O rack and their device and software versions.

Through this sensor we get the system alarm:

```
<time> PROCESS STATION <nn> FAULT IN IO RACK <i>
```
(member 'fault')

Type **'dirk'** data structure:

type

Type:	uns16
Default:	0
Description:	Type

0 = no rack in this address
1 = PIC
99 = unidentified type

version(8)

Type:	char
Default:	' '
Description:	PIC software versions

fault

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Indicates possible disturbances in PIC's operation. Specified by member 'faultind'.

Values of 'fault:binstat':
0 = no fault
1 = active fault

Time label is the time when the fault appeared ('fault'=1) or disappeared ('fault'=0).

faultind

Type:	intsev
Default:	48 0 0 0 0 0 0 1 1 1 00 14
Description:	Specifies a fault.

The values of 'faultind:intsstat:s':
0 = no fault
1 = failures on redundant bus
2 = failures on own bus
3 = PIC does not respond on redundant bus
4 = IPU faulty
5 = PIC does not respond on own bus
6 = memory fault
7 = system fault
8 = PIC does not respond
9 = no connection to HART device
10= HART configuration error

Only the most severe fault (with highest index) is indicated. Time label is the time when the fault appeared.

iocard16)

Type:	iocard
Default:	
Description:	Vector of the type ’iocard(16)’, indicating the plug−in units in the rack. For more details see the next page.

faultsum

Type:	bin
Default:	48
Description:	A sum of the I/O−plug−in units fault data.

Examples:

Example system alarm:

```
12:53:47.60 ** SYS AP01 PROCESS STATION 1 I/O DISTURBANCE, RACK 01

532a% p v :e:di:AP01:slot02:iorack01

        Print Variable

    Lid AP01 found.

 IS dirk

   MEMBER IS type IS uns16 <1><0x1>
   MEMBER IS version(8) IS char ″ 3.11 ″
   MEMBER IS fault IS binev
      MEMBER IS binstat IS bin IS uns16 <1><0x1>    1 = active fault
      MEMBER IS bintime IS time <Thu Nov 16 12:53:47.602 1995 v−64>
   MEMBER IS faultind IS intsev
      MEMBER IS intsstat IS ints <8> <0x8>    8 = PIC does not respond
      MEMBER IS intstime IS time <Thu Nov 16 12:54:22.374 1995 v−64>
   MEMBER IS iocard(16) IS iocard
   (1)
      MEMBER IS type IS uns16 <6><0x6>
    .
    .
    .
    .
```

The type representing I/O−plug−in units is ’iocard’. Through the type ’iocard’ member ’fault’ we get the system alarm:

```
<time> PROCESS STATION <nn> FAULT IN IO UNIT <i> <j>
```

Type **’iocard’** data structure:

type

Type:	uns16
Default:	0
Description:	Indicates the type of the plug−in unit plugged in the card slot

0 = EIU1
1 = EOU1
2 = BIU8
3 = BOU8
4 = AIU1
5 = AOU1
6 = AIU8
7 = FIU1
8 = AOU4

9 = TIU6
10 = PLU2 (PLUC)
11 = PLU1 (PLUN)
18 = TCU4
21 = ACU1 (ACU)
22 = ACU2 (ACUC)
23 = AIU4
26 = BIU4
27 = BIU8N
28 = AIE2
29 = AOE2
30 = BIE4
42 = MCN
43 = BIUMC
44 = BOUMC
45 = AIUMC
46 = AOUMC
47 = MBIU
48 = MBOU
54 = BIR
55 = BOR
56 = AIR
57 = AOR
60 = TIR
63 = FIR
67 = AIF
68 = AOF
89 = POR
95 = AOH
96 = unidentified plug−in unit type
97 = AIH
98 = unused card slot
99 = AIHD
100 = BIC12
101 = BICL
102 = BOCL
103 = AICL
104 = AOCL
105 = FICL
114 = DI8
115 = DO8
116 = AI8
117 = AO4
118 = AI8H
119 = AO4H
120 = FI4
122 = DI8C
123 = TI4
124 = DI8M
125 = DI8MC
127 = TC8
128 = TII4
32768 = DO8P
32769 = DO8N
32770 = DO8RO

```
32771 = DO8RC
32772 = DO8SO
32773 = DOI4RO
32774 = DOI8RO
32775 = DO16P
32777 = DOI8IRO
34832 = DI8P
34833 = DI8N
34834 = DI8U
34836 = DII8P24
34837 = DII8P48
34838 = DII4U120
34839 = DII4U125
34840 = DII4U240
34841 = DII8U120
34842 = DII8U240
34843 = DI16P
34845 = DIO32
34846 = DII8U48
34848 = DII16MN
34849 = DIA4
35091 = DI8M
35100 = DI8MN
36896 = AO4C
36897 = AO4V
36898 = AO4H
36899 = AO4DV
36900 = AOI4C
36901 = AOI4H
36902 = AO8C
36903 = AOI8C
36904 = AOI8H
36905 = AOA2H
38960 = AI8C
38961 = AI8V
38962 = AI8H
38963 = AII8CN
38964 = AII8C
38965 = AII8V
38966 = AII4H
38967 = HC8
38968 = AI8CN
38969 = AIF8V
38970 = AIF8T
38971 = AI2B
38972 = ESP8C
38973 = AI16C
38974 = AII16C
38975 = AII16CI
38976 = AII16H
38977 = AII8H
38978 = AIA2H
38979 = HC8S
38980 = AII12CI
41024 = TI4W3
```

41025 = TI4W4
41026 = TII4W3
41027 = TII4W4
41056 = TCI8
45136 = FI4V
45137 = FI4S5
45138 = FI4S24
45139 = FII4V
47232 = MST4
49268 = AOI3S
51312 = AIT4C
51313 = AIT4L
51314 = AIF4E
51315 = AIF4V
51573 = OSP3−S
51574 = OSP3−S

version(8)

Type:	char
Default:	’ ’
Description:	Indicates the software version of the plug−in unit in the card slot.

fault

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 00 14
Description:	Indicates possible disturbances in the plug−in units operation.

Specified by member ’faultind’. Values of ’fault:binstat’:
0 = no fault
1 = active fault

Time label is the time when the fault appeared (’fault’=1) or disappeared (’fault’=0).

faultind

Type:	intsev
Default:	48 0 0 0 0 0 0 1 1 1 00 14
Description:	Specifies a fault.

The values of ’faultind:intsstat:s’:
0 = no fault
1 = unit does not respond on redundant bus
2 = unit does not respond on own bus
3 = line fault in one or more channels
4 = auxiliary power overload in one or more channels
5 = wrong channel type
6 = memory fault
7 = system fault
8 = unit does not respond on any bus
9 = no connection to the HART device
10 = error in HART configuration

Only the most severe fault (with highest index) is indicated. Time label is the time when the fault appeared.

chann

Type:	uns32
Default:	0
Description:	Indicates what plug−in units channels are in use (= in I/O module FBC).

The channels are coded in separate bits so that bit 0 is channel 0, bit 1 channel 1, and so on, until 31.

The bit values are coded as follows:
0 = unused channel (I/O module not configured)
1 = used channel (I/O module configured)

channfault

Type:	uns32
Default:	0
Description:	Indicates what plug−in units channels have channel−specific faults (line fault, auxiliary voltage overload, wrong unit type).

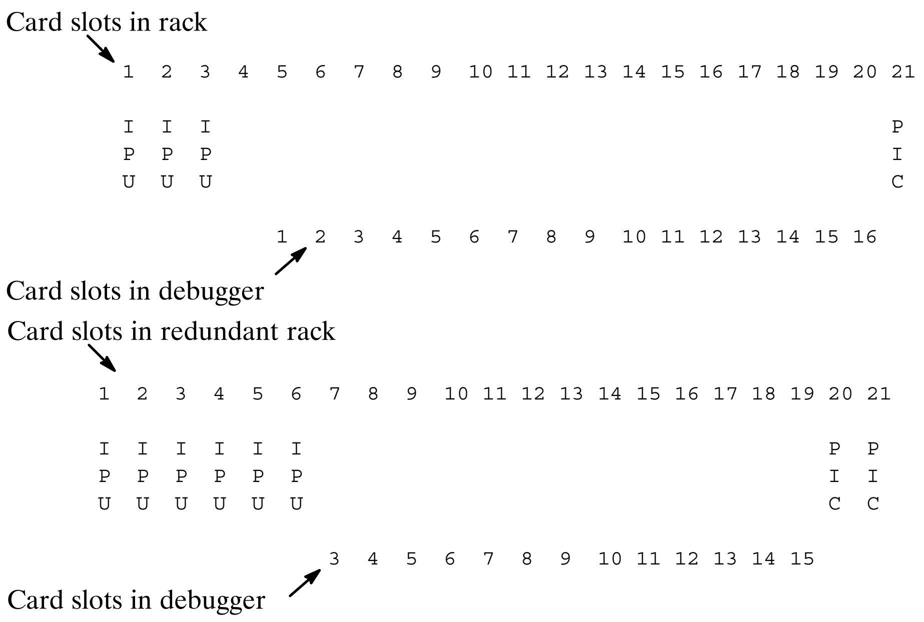
The channels are coded in separate bits so that bit 0 is channel 0, bit 1 channel 1, and so on, until 31.

The bit values are coded as follows:
0 = channel OK
1 = channel faulty

Example system alarm:

```
12:43:07.11 ** SYS AP01 PROCESS STATION 1 I/O DISTURBANCE, CARD 01
05
```

Card slots are given as follows:



```
Card slots in rack

    1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21

    I   I   I                                                                       P
    P   P   P                                                                       I
    U   U   U                                                                       C

                    1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16

Card slots in debugger
Card slots in redundant rack

    1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21

    I   I   I   I   I   I                                                       P   P
    P   P   P   P   P   P                                                       I   I
    U   U   U   U   U   U                                                       C   C

                            3   4   5   6   7   8   9   10  11  12  13  14  15

Card slots in debugger
```

```
533a% p v :e:di:AP01:slot02:iorack01:iocard(01)

        Print Variable
    Lid AP01 found.
 IS iocard
   MEMBER IS type IS uns16 <6><0x6>     6 = AIU8
   MEMBER IS version(8) IS char " 3.0  "
   MEMBER IS fault IS binev
      MEMBER IS binstat IS bin IS uns16 <1><0x1>     1 = active fault
      MEMBER IS bintime IS time <Thu Nov 16 12:43:07.113 1995 v-64>
   MEMBER IS faultind IS intsev
      MEMBER IS intsstat IS ints <5> <0x5>     5 = wrong channel type
      MEMBER IS intstime IS time <Thu Nov 16 12:42:02.554 1995 v-64>
   MEMBER IS chann IS uns32 <67><0x43>     used channels 0,1,6
   MEMBER IS channfault IS uns32 <3><0x3>     faulty channels 0,1
```

Chann and channfault (Below is not valid on ACU and PLU units):

```
ch      7   6   5   4    3  2  1  0
value   8   4   2   1    8  4  2  1     hex
value  128  64  32  16   8  4  2  1     des
```

Example:

Chann = 67 and channfault = 3

Used channels:

```
ch      7  6   5  4   3  2  1  0
value   0  4   0  0   0  0  2  1 = 43 hex
value   0  64  0  0   0  0  2  1 = 67 des
```

Faulty channels:

```
ch      7  6  5  4   3  2  1  0
value   0  0  0  0   0  0  2  1 = 3  hex
value   0  0  0  0   0  0  2  1 = 3  des
```

6.11 Application Server’s Update State in the Backup Server (from the Backup Server’s Point of View)

Updating the runtime data of the modules from the Application Server to the Backup Server is described from the Backup Server’s point of view with a sensor whose name is as follows:

```
:e:di:<backup server name>:<application server>:backup
```

where <application server> is the Application Server identifier wanted. The sensor’s type is ’**stdb**’.

Saving of configuration and data to the Backup Server affects the sensor’s data.

Type **’stdb’** data structure:

updvol

Type:	intl
Default:	48 0
Description:	Update frequency of module collection, updates/h

lastsccs

Type:	time
Default:	0 0 0 0 0 1 1 1 0 14
Description:	Time of module collection’s last update

upderr

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 0 14
Description:	Update state that indicates whether the data update of the Backup Server causes a fault in the Application Server.

Data ’upderr:binstat’:
0 = no fault
1 = active update fault

module(64)

Type:	char
Default:	
Description:	The first module without updates

version

Type:	char
Default:	
Description:	Module’s version

6.12 Application Server’s Update State in the Backup Server (from the Application Server’s Point of View)

Data updating from the Application Server to the Backup Server from the Application Server’s point of view is described with a sensor whose name is as follows:

```
:e:di:<application server name>:backup
```

The sensor’s type is ’**dbup**’.

Saving of configuration and data to the Backup Server affects the sensor’s data.

Type **’dbup’** data structure:

datesize

Type:	intl
Default:	48 0
Description:	Total data size for cyclic updates

daterate

Type:	intl
Default:	48 0
Description:	Cyclic update rate [min]

datecount

Type:	intl
Default:	48 0
Description:	Count of cyclic updates since Application Server start−up

lastsccs

Type:	time
Default:	0 0 0 0 1 1 1 0 0 14
Description:	Time of last successful backup

upderr

Type:	binev
Default:	48 0 0 0 0 1 1 1 0 0 14
Description:	Update state that indicates whether the data update of the Backup Server causes a fault in the Application Server.

Data ’upderr:binstat’:
0 = no fault
1 = active update fault

errind

Type:	intsev
Default:	48 0 0 0 0 0 1 1 1 0 0 14
Description:	Fault index which indicates the type of update fault

Data ’errind:intsstat:s’:
0 = no fault
1 = module difference between application and Backup Server module collections
2 = Backup Server does not receive updates
3 = no connection to Backup Server
4 = system fault

module(64)

Type:	char
Default:	
Description:	Module that cannot be updated

version(16)

Type:	char
Default:	
Description:	Module’s version.

6.13 Backup Server’s Control

Backup Server’s control is described with a sensor whose name is as follows:

```
:e:sn:<application server name>:bup_ctr
```

The sensor’s type is **’buct**’.

Type **’buct’** data structure:

daterate

Type:	int
Default:	48 0
Description:	Data update period/min

0 = no update

datevent

Type:	binev
Default:	48 0 0 0 0 1 1 1 0 0 14
Description:	Application Server’s backup state

Data ’dataevent:binstat’:
0 = waiting update time
1 = update active

command

Type:	intl
Default:	48 0
Description:	Backup command

operation

Type:	intl
Default:	48 0
Description:	Active command

6.14 Backup Server’s Database

The database which is located in the Backup Server is represented as a sensor whose name is as follows:

```
:e:di:<backup server name>:database
```

The sensor’s type is **’budb**’.

Saving of configuration and data to Backup Server affects the sensor’s data.

Type **’budb’** data structure:

dbcount

Type:	ints
Default:	48 0
Description:	Number of updated Application Server databases.

updvol

Type:	intl
Default:	48 0
Description:	Total update frequency modules/min

updderr

Type:	binev
Default:	48 0 0 0 0 0 1 1 1 0 14
Description:	Update state that indicates whether the data update of the Backup Server causes a fault in any Application Server.

Data ’upderr:binstat’:
0 = no fault on any Application Server
1 = active update fault on some Application Server

dbname(48)

Type:	char
Default:	
Description:	Name of database containing modules which have not been updated

module(64)

Type:	char
Default:	
Description:	Module without updates.

version(16)

Type:	char
Default:	
Description:	Module’s version.

6.15 Licence Description

The amount of Process Control Server norm−I/O is monitored with a sensor whose name is:

```
e:di:<application server name>:authority:NormCapacity
```

The licence type of the licence request rejected in the Process Control Server is released in a sensor whose name is:

```
e:di:<application server name>:authority:licence
```

The type of all sensors is **’dlic’**. The sensors contain information on the specific licence type.

The data structure for type **’dlic’**:

expire

Type:	time
Default:	
Description:	Licence’s expiration date

liclimit

Type:	intsev
Default:	
Description:	Maximum number of licences

Tells the maximum number of licenses in the Process Control Server which are of the type in question. Value and time stamp are set when the sensor is created.

licfree

Type:	intsev
Default:	
Description:	Number of free licences

The value tells how many licenses of the type are free. Value and time stamp are set when the sensor is created and they are updated when the license of the type is reserved or freed.

licrejected

Type:	intsev
Default:	
Description:	Number of rejected licences

The value tells cumulatively how many requests for license of the type have been rejected. The value increases by one each time the license server rejects a request for a license. Time stamp is updated at the same time.

rejectonline

Type:	intsev
Default:	
Description:	On−line command rejected

0 = no rejected command
1 = add on−line command rejected
2 = swap on−line command rejected

errmod1

Type:	txt84
Default:	
Description:	Function without licence

The value tells the name of the rejected license request.

- In the case of function blocks, the name is the module of the function block.
- In the case of the OPS window license, the name is the IP address of the display that requested for a window.
- In the case of the rejecting the capacity the name is the module name that will not be loaded.

If the rejected module's name is already in the sensor, it will not be logged again. If the sensor has got the module name when the module is accepted to the Process Control Server, the name is deleted from the sensor.

errmod2

Type:	txt84
Default:	
Description:	Function without licence

See errmod1.

errmod3

Type:	txt84
Default:	
Description:	Function without licence

See errmod1.

errmod4

Type:	txt84
Default:	
Description:	Function without licence

See errmod1.

errmod5

Type:	txt84
Default:	
Description:	Function without licence

See errmod1.

fault

Type:	binev
Default:	
Description:	Fault in authorization handling

The value tells if there has been a fault in the license type’s license processing or there is a function without a licence.

0 =	no fault
1 =	fault.

The time stamp is updated only when the value is changed.

faultind

Type:	intsev
Default:	
Description:	Authorization handling fault index

0:	no fault
1:	The CPU configuration file does not have the license data required by the license type (unlock code missing)
2:	there have been more requests than there are assigned licenses (too many requests)
3:	the license data in the CPU configuration file is incorrect (unlock code rejected)
4:	the approved limit of the Norm I/O has been exceeded (capacity exceeded)
5:	the approved limit of the Norm I/O has been exceeded (capacity warning)
6:	A second license data, with less licenses defined, for the same license type has been found in the CPU configuration file (less licenses defined)
7:	out of memory
8:	unknown license type requested in the Process Control Server
9:	license type expired
10:	license type expire warning

The time stamp is updated only when the value is changed.

6.15.1 Example Printout

The following is an example output of a sensor describing the Process Control Server norm I/O capacity.

```
7481a% p v :e:di:UP01:authority:NormCapacity
        Print Variable
    Lid UP01 found.
 IS dlic
   MEMBER IS expire IS time <        00 00:00:00.000 2000 v00>
   MEMBER IS liclimit IS intseev
      MEMBER IS intsstat IS ints <3000> <0xbb8>
      MEMBER IS intstime IS time <Sun Aug 20 07:48:30.151 2017 v32>
   MEMBER IS licfree IS intsev
      MEMBER IS intsstat IS ints <1733> <0x6c5>
      MEMBER IS intstime IS time <Mon Aug 21 06:16:20.329 2017 v32>
   MEMBER IS licrejected IS intsev
      MEMBER IS intsstat IS ints <0> <0x0>
      MEMBER IS intstime IS time <Sun Aug 20 07:48:30.151 2017 v32>
   MEMBER IS rejectonline IS intsev
      MEMBER IS intsstat IS ints <0> <0x0>
      MEMBER IS intstime IS time <Sun Aug 20 12:04:34.353 2017 v32>
   MEMBER IS errmod1 IS txt84(84) IS char <null>
   MEMBER IS errmod2 IS txt84(84) IS char <null>
   MEMBER IS errmod3 IS txt84(84) IS char <null>
   MEMBER IS errmod4 IS txt84(84) IS char <null>
   MEMBER IS errmod5 IS txt84(84) IS char <null>
   MEMBER IS fault IS binev
      MEMBER IS binstat IS bin IS uns16 <0><0x0>
      MEMBER IS bintime IS time <Mon Aug 21 06:16:20.329 2017 v32>
   MEMBER IS faultind IS intsev
      MEMBER IS intsstat IS ints <0> <0x0>
      MEMBER IS intstime IS time <Mon Aug 21 06:16:20.329 2017 v32>
```

The capacity for the Process Control Server has been defined as 3000 norm I/O. At the moment of the output, the Process Control Server has 1733 norm I/O free capacity. After the Process Control Server has been started the capacity limit (3000 norm I/O) has not been exceeded, and no modules are left unloaded to the Process Control Server due to exceeding the norm I/O limit. No fault situations have occurred during the Norm I/O licenses processing.