

Remote Monitor and Control Protocol (RMCP)

For NTC/21xx and NTC/22xx Equipment.

Usermanual

Table of Contents

1	Release history.....	3
2	What is RMCP?.....	4
3	RMCP v1.0 versus v2.0.....	4
4	Serial RMCP.....	5
4.1	Connecting to the device.....	5
4.2	Line Settings.....	5
4.3	Setting up the device for serial M&C.....	6
4.3.1	Serial M&C interface type - SyDevSerIfType.....	6
4.3.2	Device address for serial interface - SyDevRs485Addr	6
4.3.3	Serial interface baudrate - SyDevBaudrate	7
5	Ethernet RMCP	8
5.1	Setting up the device for M&C over Ethernet.....	8
5.1.1	Device IP address - SyDevIPAddr.....	8
5.1.2	Device IP mask - SyDevIPMask	8
5.1.3	Default gateway IP address - SyDevIPGateWay.....	8
5.1.4	Ethernet M&C interface type - SyEthTransType.....	9
5.1.5	Device MAC address - SyDevMacAddr.....	9
6	RMCP Protocol.....	10
6.1	Message Format	10
6.1.1	Start Byte	10
6.1.2	Address Byte.....	10
6.1.3	Message Header.....	11
6.1.4	Message Data.....	11
6.1.5	End of Text Byte	11
6.1.6	Checksum Byte.....	11
6.1.7	Example of checksum byte calculation.....	11
7	Error handling.....	12
8	Logical addressing.....	13
8.1	Example of a device with multiple functions	13
8.2	Example of a drawer with multiple devices	13
9	Command Classification.....	14
9.1	Standard Commands (or vars).....	14
9.1.1	Examples of standard command	14
9.2	Special commands.....	15
9.2.1	Example of a special command.....	15
9.2.2	Example of a structured command	16
9.2.3	Example of an array command.....	17
10	RMCP Loader.....	18
11	Some hints on implementation of RMCP	19
11.1	Alarm handling.....	19
11.2	Dynamic alarm string.....	19
11.3	Sync bytes	19
11.4	A practical example of alarm handling	20
12	Questions and answers.....	22
13	Some comparison notes between the NTC/21xx and NTC/22xx series.....	25

Tables

Table 1	Release history manual	3
Table 2	RS485 serial interface	5
Table 3	RS232 serial interface	5
Table 4	Error reply codes	12
Table 5	Standard commands variants.....	14

1 Release history

Table 1 Release history manual

Version	Date	Details
v0.0	May 18, 2005	Start of document
v1.0	May 19, 2005	First internal release of general RMCP manual
v1.1	August 10, 2005	Added an example of the implementation of alarm handling
v1.2	December 2, 2005	Added questions and answers section, reflecting on some of the re-occurring questions from customers
V1.3	December 10, 2005	Added some comparison notes between NTC/21xx and NTC/22xx series in answer to some specific questions of customers
V1.4	February 26, 2007	Chapter 7 – Error Handling New error reply codes added in the table
V1.5	August 10, 2007	Chapter 7 – Error Handling update New error reply codes added in the table
V1.6	September 10, 2007	Chapter 7 – Error Handling update New error reply codes added in the table

2 What is RMCP?

RMCP stands for Remote Monitor and Control Protocol which is a message oriented protocol.

This manual explains how Newtec devices can be remotely monitored and controlled via the serial port or via Ethernet. It is a general manual intended for all possible NTC/21xx and NTC/22xx equipment. A separate manual is also delivered in html format that lists all the commands that are known to a specific model (for example the NTC/2163 DVB demodulator). This html file has the same structure as known to the equipment and contains the description as provided in the operator usermanual.

3 RMCP v1.0 versus v2.0

- The NTC/20xx series of equipment used RMCP commands as defined in RMCP v1.0 to provide M&C functionality. RMCP v1.0 commands are listed in the operational usermanual.
- During the development of the NTC/21xx series the RMCP platform was extended to offer more flexibility (...). This resulted in RMCP v2.0. For reasons of backward compatibility the NTC/21xx series was able to operate in both RMCP v1.0 and v2.0. The RMCP v1.0 commands can be requested; in that case the corresponding operational user manual will be used. For RMCP v2.0 a separate user manual can be requested. This manual is automatically generated from the databases that define the features of equipment. It consists of a general manual that is the same for all equipment and an html manual that has a description of all possible commands for the software that runs on specific equipment.
- For the NTC/22xx series it was decided to migrate completely to RMCP v2.0.

4 Serial RMCP

4.1 Connecting to the device

The serial monitor and control port can be switched between RS232 and RS485, by default RS485 is selected. RS485 is typically used when more than 1 device needs to be controlled with a parallel drop-down cable.

Table 2 RS485 serial interface

Pin	Name	Function
1	GND	Shield ground
2		Not connected
3	Tx-A	Send Data A (input)
4	Rx-A	Receive Data A (output)
5	GND	Signal ground.
6	Rx-B	Receive Data B (output)
7		Not connected
8		Not connected
9	Tx-B	Send Data B (input)

Table 3 RS232 serial interface

Pin	Name	Function
1	GND	Shield ground
2	Rx-D	Receive Data (input)
3	Tx-D	Transmit Data (output)
4	DTR	Data Terminal Ready (output)
5	GND	Signal ground.
6		Not connected
7	RTS	Request To Send (output)
8	CTS	Clear To Send (input)
9		Not connected

4.2 Line Settings

The main line settings for this serial interface are:

- asynchronous data transfer
- 1 start bit (logic "0")
- 7 data bits (LSB first on line)
- even parity
- 1 stop bit (logic "1")
- 4800, 9600, 19200, 38400, 57600 or 115200 baud

There is no flow control on the serial interface. Next to correct formatted messages, the only significant character is the SYNC-character (value 16 hex), which is sent by the device to indicate that it is busy executing the command and preparing the response. This prevents other devices from taking control of the bus if the response cannot be given immediately.

4.3 Setting up the device for serial M&C

In the ../Unit/Setup/Serial port settings following parameters can be set:

4.3.1 Serial M&C interface type - SyDevSerIfType

```
NTC2163/Unit/Setup/Serial port settings
Serial interf. type: RS485
```

description: M&C serial port interface type, RS485 (default) or RS232.
RS232 is used for M&C of a single device , RS485 is typically used for multiple devices on a single bus.

access level: expert only

selections:

- RS485
- RS232

4.3.2 Device address for serial interface - SyDevRs485Addr

```
NTC2163/Unit/Setup/Serial port settings
Device RMCP address: 50
```

description: The device address, used in the messages for remote serial M&C, is a single byte with a value in the range 49 (31 hex - ASCII "1") up to 110 (6E hex - ASCII "n"). It identifies the device that has to handle the message from the remote control unit.

When the multi-user RS485 bus is used, each device on the bus must have a different address, unique in the system.

Address 111 (6F hex - ASCII "o") is the "broadcast" address. This can be used when only one device is connected to a COM-port of a PC to address the device without knowing its exact address.

range: 49/110

4.3.3 Serial interface baudrate - SyDevBaudrate

NTC2163/Unit/Setup/Serial port settings
Serial baudrate: 115200

description: Interface baudrate for serial monitor and control via the RMCP protocol.

selections:

- 4800
- 9600
- 19200
- 38400
- 57600
- 115200

5 Ethernet RMCP

RMCP commands can be sent to the device using the Ethernet interface. The commands are sent as data in a TCP/IP stream. The used socket number is 5933. The RMCP protocol is exactly the same as for the serial interface, with one small exception: The RMCP address of the device (that is present in an RMCP command) will be ignored by the receiving device. To enable the device to communicate over Ethernet, the Ethernet interface needs to be configured. This is done by setting the IP address and net mask using the keyboard interface.

5.1 Setting up the device for M&C over Ethernet

In the ../Unit/Setup/Ethernet settings following parameters can be set:

5.1.1 Device IP address - SyDevIPAddr

```
NTC21xx/Unit/Setup/Ethernet settings
Device IP address: 192.168.254.1
```

description: Device IP address.

5.1.2 Device IP mask - SyDevIPMask

```
NTC21xx/Unit/Setup/Ethernet settings
Device IP mask: 255.255.255.0
```

description: Device IP mask.

5.1.3 Default gateway IP address - SyDevIPGateWay

```
NTC21xx/Unit/Setup/Ethernet settings
Default gateway: 192.168.254.206
```

description: IP address of default gateway.

5.1.4 Ethernet M&C interface type - SyEthTransType

NTC21xx/Unit/Setup/Ethernet settings
Ethernet interface: TCP

description: Selection of the Ethernet interface transport layer, TCP (default) uses acknowledges that confirm reception of messages while UDP does not.

UDP has the advantage of being faster since it does not require the "wait for acknowledge". Furthermore RMCP over Ethernet already has protection on the RMCP layer by means of the CRC so there is no need for the extra protection provided by the TCP-type layer.

access level: expert only

selections:

- TCP
- UDP

5.1.5 Device MAC address - SyDevMacAddr

NTC21xx/Unit/Setup/Ethernet settings
Device MAC address: 00:06:39:00:10:5D

description: Device MAC address. Read-only.

6 RMCP Protocol

The control unit sends a “request” message to a device, identified by its unique address. The addressed device interprets the message, performs the requested action and sends back a “response” message.

The receiving device rejects all messages with transmission errors without any further action. Transmission errors are: no stop bit, parity error, LRC-error and message receive buffer overflow.

All correct formatted messages, except some special system messages, are responded by the addressed device with an acknowledge message. Only in a few restricted cases, the device does not respond to a request from the control unit. This is for example the case when a general device reset is requested.

Correctly received messages, which can not be handled by the device, are refused via a no-acknowledge “error” message, containing the reason why the message is rejected.

A device never sends messages on its own initiative. It only responds to a request from the control unit. The total transmit time of a complete message may not exceed 250 ms. If the message is not completed within this time it is discarded.

6.1 Message Format

The general syntax for all messages is :

- start byte
- address byte
- message header
- message data
- end of text byte
- checksum byte

6.1.1 Start Byte

The start byte defines unequivocal the beginning of a new message.

There are three possible start bytes:

02hex :start-of-text (STX), used for a request send by the control unit to a device
06hex :acknowledge (ACK), used for the response from a device to the control unit
15hex: no-acknowledge (NAK), used for the “error” response from a device to the control unit.

6.1.2 Address Byte

The address identifies the destination device for the request message and the source device for the response message. Each device must have a unique address when the control unit guards more than one device via a multi-user bus.

Valid addresses are in the range 49 (31hex, ASCII “1”) up to 109 (6Dhex, ASCII “m”).

The indoor and outdoor units are two different logical devices for RMCP, although they communicate via the same electrical link. The outdoor unit has the address of the indoor unit raised by one.

For a limited set of messages, the RMCP-broadcast address 111 (6Fhex, ASCII “o”) can be used. This address is reserved for (test) systems with only one device connected to the control unit. Messages that can be handled with the broadcast address are the device identification and the RMCP-address.

6.1.3 Message Header

The header defines the contents of the message and thus how the device should handle it. All headers consist of three capital or minor ASCII characters (respectively 41_{hex} up to 5A_{hex} and 61_{hex} up to 7A_{hex}). The header is always followed by !, ? or ^ (set, get and get variable specifications). The header syntax is case sensitive.

6.1.4 Message Data

The message text is a string of ASCII representable characters (20_{hex} to 7F_{hex}). The header defines the meaning and size of this string.

Possible message text is:

- no data, this is indicated as the "null"-string
- an enumeration, selected from a limited set of alternatives
- a numerical value, with the format and units as given for each command. Format and size are fixed. Example "s##.#" for sign, two digits, decimal dot and one digit
- a text string, with fixed or variable length
- a dummy argument, this is a string of question marks that replaces the normal value when the requested parameter is unknown
- a not applicable indication, this is a string of "#"-signs, which replaces the normal value when the requested parameter is not applicable.

6.1.5 End of Text Byte

The end of text byte indicates the end of the message text. This is the "end-of-text" character, with as value 03 hex. This is the last but one byte of the complete message. It is only followed by the checksum.

6.1.6 Checksum Byte

The last byte of each message is a checksum. This is a one-byte longitudinal redundancy check (LRC). It is calculated as the bit wise exclusive-OR of all the message bytes, including the message delimiters (STX, ETX,) and the address.

The bit wise exclusive-OR function can be calculated by writing the binary value of all the concerned bytes beneath each other and then performing a modulo-2 addition on each column, thus without carry propagation (see examples).

6.1.7 Example of checksum byte calculation

For a unit with address 51 (33_{hex}, ASCII "3")

<STX>3TTt1<ETX>w			
<STX>	02 hex	000 0010	
3	33 hex	011 0011	
T	54 hex	101 0100	
T	54 hex	101 0100	
t	74 hex	111 0100	
1	31 hex	011 0001	
<ETX>	03 hex	000 0011	
<LRC>	77 hex	111 0111	ASCII "w"

7 Error handling

In case there is an error in the RMCP message the device will reply with an error number:

Table 4 Error reply codes

Number	Description
000	Argument in range
004	Failed to backup variables, the backup pool too small (Special Errors only for software debugging)
005	Failed to restore Variables, the backup restore was failed (special errors only for software debugging)
023	Header unknown
024	Operation is unknown (!,?,^)
025	Only basic RMCP commands are allowed (blocking mode)
026	Operation does not exist for this command (Read/Write)
027	Command suppressed by SW/HW Capability
028	Access denied by user
029	Suppressed due to other application settings
030	Format request error: unknown identifier
031	Format Request Error: No Set or Get Variant
032	Format Request Error: Var index error
034	Data syntax error
035	String format error
040	Routing Not Available
041	Routing Timeout Error
042	Routing Failed
110	Argument not acceptable, wrong enumeration
111	Argument out of range (enum)
112	Argument too low (numeric)
113	Argument too high (numeric)
200	Get failed
300	New setting was failed and the original setting (backup) failed too
301	New setting was failed but the original Setting (backup) is active again
302	Setting was rejected due to other system setting(s)

8 Logical addressing

An equipment can contain multiple devices and a device can contain multiple identical functions. To address a function in a specific device in an equipment, one has to use logical addressing. A logical address is a sub address and simply contains the device id and the function id:

syntax: ":device id:function id" with

-device id: one or more ASCII character

'0' (0x30) : the common part, to address common functions (e.g. IP address, RMCP mode, drawer serial number..)

'1' (0x31) : first device (e.g. a modulator)

'2' (0x32) : second device (e.g. a demodulator)

...

-function id: one or more ASCII character

'1' (0x31) : first function

'2' (0x31) : second function

..

(e.g. function 17 in device 2 is :2:17)

When no logical address is used in the RMCP message, address ":1:1" is assumed. So when a device has no multiple functions or devices, no logical address is needed.

8.1 Example of a device with multiple functions

A NTC/2142 (IF/L-Band Frequency Agile Up/Down converter) can have 2 identical functions 'ODU power supply enable', one for the installed up converter conditioner and one for the installed down converter conditioner; an example will make this clear.

:1:1ODp!1 (enables the power supply voltage at the output of the up converter conditioner)

:1:2ODp!1 (enables the power supply voltage at the input of the down converter conditioner)

8.2 Example of a drawer with multiple devices

A NTC/21xx modem holding a modulator and a demodulator board can common functions (e.g. symbol rate); an example will make this clear:

:1:1TRS? (gets the symbol rate of the modulator board)

:2:1TRs? (gets the symbol rate of the demodulator board)

9 Command Classification

All RMCP (remote monitor and control protocol) commands consist of 3 characters, followed by !, ? or ^ (set, get and get variable specifications). Note that RMCP commands are case sensitive.

In this device we distinguish 2 classes of RMCP commands; standard commands and special commands.

9.1 Standard Commands (or vars)

- Map directly onto a state variable with permanent storage.
- Are completely defined by a single component definition file.
- Exist in 3 variants, determined by the data following the 3-character RMCP-header :

Table 5 Standard commands variants

Standard Command variants				
Command Body	Command Variant	Function Call?	Reply	Description
?	Get VarValue	No	value	Returns current value of the variable in it's proper format.
!	Set VarValue	Yes	same as argument	set variable to new value and call function for further dedicated processing.
^	Get VarSpec	No	spec	Returns the specifications of the variable.

9.1.1 Examples of standard command

```

send:      SLs?           // get serial number of device
reply:     SLs?01052433  // serial number = 01051433

send:      Adr!100      // set device address to 100
reply:     Adr!100      // address is set to 100

send:      DSM!1        // set sleep mode to sleep
reply:     DSM!1        // sleep mode is set to sleep

send:      SLs^         // get specifications (syntax) of the serial number
reply:     SLs^<"%s",8"> // syntax is 8 integers

```

9.2 Special commands

- Have no one-to-one mapping to state variables but have dedicated implementations.
- Can have different receive and reply argument lists.

The vars they refer to can be either state vars or temporary vars. Temporary vars storage only exists during the processing of the command.

For reasons of uniformity, special commands also exist in 'set' and 'get' versions but the command/reply argument lists can differ between the get and set version.

9.2.1 Example of a special command

1.2.1.1 Device mode – SyDevMode

description:

The following device operating modes are defined:

Normal mode: This is the standard operating mode which enables the default set of parameters that are most frequently used.

Expert mode: This mode gives an "expert" operator access to an additional set of more advanced parameters.

A password is required to switch to expert mode. This password is the model number of the device (e.g. 2180 for a NTC/2180 DVB L-band modulator) or the software identification number in case of a board (e.g. 6161 for a DVB-S modulator board).

Commands that can only be changed in expert mode are indicated in this user manual by "access level: expert only".

rmcp header:

SMm (get and set)

get elements:

get reply elements:

[SyDevModeState](#)

set elements:

[SyDevModeState](#) , [SyDevModePass](#)

set reply elements:

[SyDevModeReply](#) , [SyDevModeState](#)

device id:

0

Example:

```

send:          SMm?           // get current operation mode of device
reply:        SMm?1          // current device mode is normal (1)

send:          SMm!2,2180     // set device mode to expert (2), using password 2180
reply:        SMm!0,2        // success (0), now operation mode is expert (2)

```

9.2.2 Example of a structured command

A structured command can have more than 1 argument that are separated by commas ','. These are especially useful to get or set parameters that are closely related. In fact a structured command is a special case of a special command where get and set replies have the same format and consist of an equal number of arguments.

1.3.2.8 PRBS counter – BecBec

description:

Reads the current status (PRBS synchronisation), elapsed time since start of the PRBS counter being locked and the number of PRBS errors counted since last lock. Reset the time and number counters by pushing the "CLR" key.

rmcp header:

Bec (expert: get, normal: no access)

structure:

[BecStatus](#) , [BecErrCnt](#) , [BecSecs](#)

device id:

1

The arguments and replies of special commands are hyperlinked to their definition (in the html rmcp manual). In the explanation of the structure the elements are separated by "space" "comma" "space" for clarity. When sending the message the spaces need to be omitted so that arguments and replies are only separated by commas ",".

9.2.3 Example of an array command

An array command is used when a variable is used over a number of identical instances. In the RMCP command overview array commands have the “array range” indicated.

1.2.1.8.5 Trap community 1 - SyTrapCommunity1

description:

SNMP trap community

rmcp header:

TCO (get and set)

device id:

1

array range:

1..2

Example:

```
send:          TIP?[1]           // get first trap ip address
reply:         TIP?[1]192.168.2.169 // first trap ip address is 192.168.2.169
send:          TIP?[2]           // get first trap ip address
reply:         TIP?[2]192.168.2.171 // second trap ip address is 192.168.2.171
```

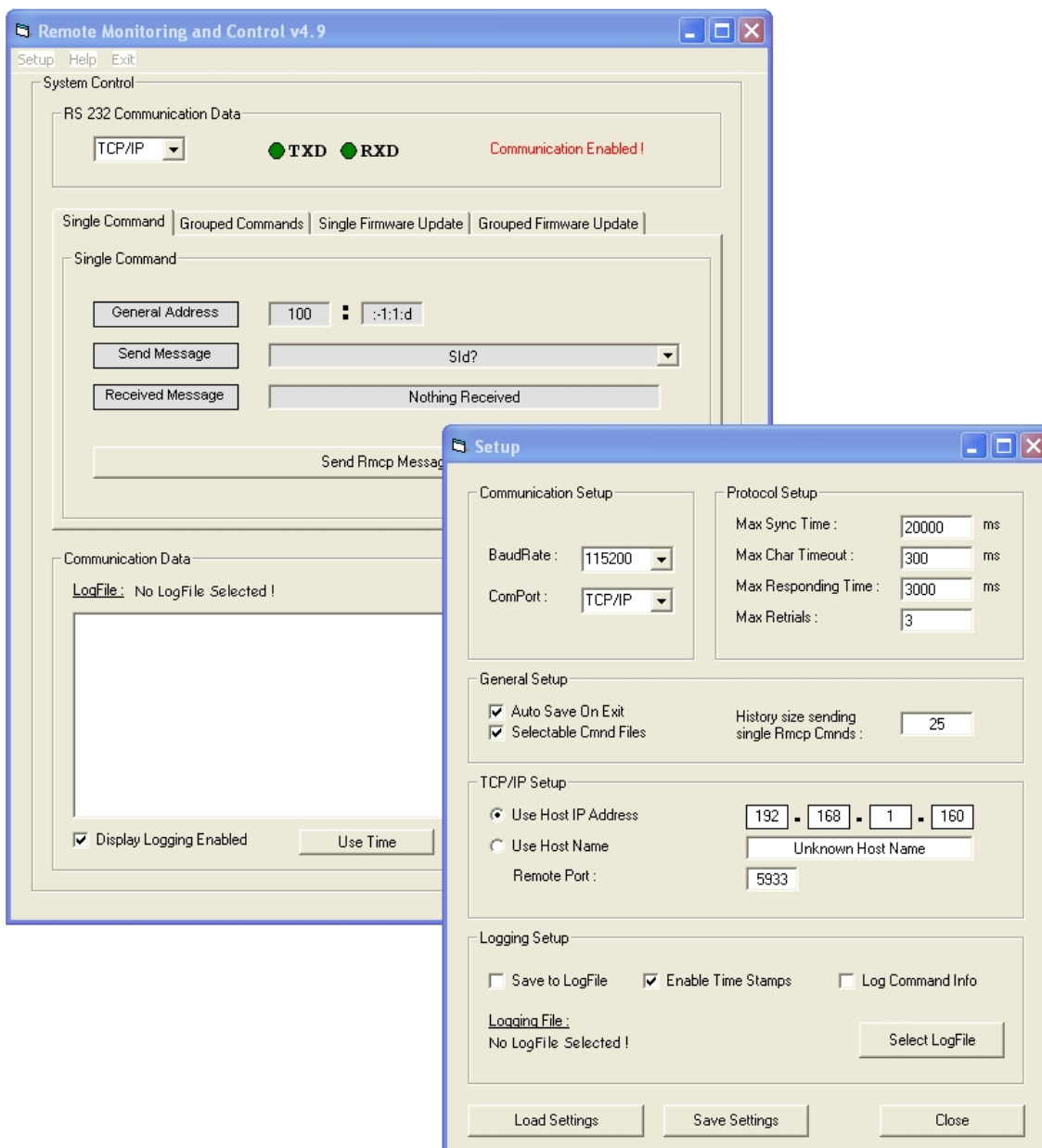
10 RMCP Loader

RMCPLoader offers a Windows platform based user interface that allows:

- Single command control
- Grouped command control
- Single firmware update
- Grouped firmware update

for any Newtec NTC/20xx or NTC/21xx device that is equipped with a serial monitor and control port or an Ethernet connection port. This program can be useful when implementing a new management system to check the behaviour of the equipment on RMCP commands.

The program and its usermanual can be downloaded freely at <http://www.newtec.be/support/download.shtml>



11 Some hints on implementation of RMCP

11.1 Alarm handling

Each 21xx or 22xx device contains information on the proper working of itself by means of alarms. A set of RMCP commands are foreseen that need to be used to:

- get information about alarms ([AIAAlarmCount](#), [AIAAlarmDesc](#))
- get the current and memorised status of the alarms ([AIAAlarmCur](#), [AIAAlarmsCur](#), [AIAAlarmMem](#), [AIAAlarmsMem](#), [AIAAlarmTime](#))
- get and set the operation mode of the alarms ([AIAAlarmMode](#), [AIAAlarmsMode](#))

The number of alarms is related to the configuration (i.e. the number of boards and their types). It is also possible that the number of alarms of a device can increase in the lifecycle of the software when new features are added. Therefore special care has to be taken when writing a management system.

A Network Management System (NMS) needs to poll the alarm string on a regular base by using the command Current alarm status string – [AIAAlarmsCur](#) (CAs). This can be done using the get version (CAs?) that will get the actual alarm string without resetting the memorised alarm indications. An alarm that was active but is not active anymore at the time the alarm string is read will still indicate alarm until it is reset. This is to make assure that momentary alarms that occur in between 2 polling cycles are not missed by the NMS.

After acknowledgement of the NMS that the alarm string is properly received, it could send the set version (CAs!) that will return the actual alarm string after which the device will reset the alarm string. This is to erase all memorised alarms. If the alarm string is read again, it will show only the active alarms.

11.2 Dynamic alarm string

The alarm string (get with CAs?) is built up dynamically depending on the hardware that has been installed in the chassis (drawer) of the device. New alarms will become available if boards are replaced (by another type) or added. Also, when additional feature become available in higher (newer) software versions, new alarms can be added to the alarm string. In order to overcome problems that occur when working with static alarm strings that translate to a fixed alarm, the use of the alarm indexing is highly recommended.

With this system, the alarm string is read (CAs? or CAs!) then scanned by the NMS for any present alarm (value = not 0). The description of that alarm can then be retrieved by using the Alarm index description - [AIAAlarmDesc](#) that returns a short textual description of the n-th alarm-buffer of the device. The reply will consist of two parts separated by a semicolon ";". The first part is the alarm name (a unique name for each alarm), the second part is a short description of the alarm. The NMS then can act accordingly when it detects a certain alarm.

This way, the management system will not need to have the same alarm bit to be present at the exact same place in the alarm string so the NMS will always act the same independent of the position of an alarm in the alarm string.

11.3 Sync bytes

Every device runs separately a small thread. This thread outputs SYNC characters e.g. every 100 ms internally. The SYNC-character (value 16hex) is sent by the device to indicate that it is busy with executing the command and preparing the response. This prevents other devices from taking the bus if the response cannot be given immediately. When a device receives an RMCP message, the thread's output is redirected to the M&C port. The device interprets the message, performs the requested action and sends back a response with the confirmation of the command. Just before sending the response, the thread's output is redirected internally.

11.4 A practical example of alarm handling

This example was taken from a typical support case.

From the diagnostics report (just point your browser to the IP address of the modulator), this is the list of alarms:

NTC2155/Alarm/Device

Device reset flag	O.K.
Self test	O.K.
Incompatibility	O.K.
General device	O.K.
Interface	Alarm Cnt: 1
Reference clock	O.K.
Device temperature	O.K.
Power supply voltage	O.K.
Input framing	O.K.
Timebase sync.	O.K.
Buffer underflow	O.K.
Buffer overflow	O.K.
Clock PLL	O.K.
Synthesiser	O.K.
RF phase lock DRO	O.K.
BISS summary	O.K.
Internal M&C module	O.K.
Interface module	O.K.
Internal modulator	O.K.
Device architecture	O.K.

As you can see there are 20 different alarms for this device.

Now I'm sending following commands through RMCP:

Nas? -> Nas?20

Indicating that there are 20 different known alarms.

When I send CAs (Current alarm status string - AIAlarmsCur) this reads the status (0 or 1) of all current device alarm buffers.

The GET version returns the current alarm buffer contents, the buffer is not reset.

The SET version returns the current alarm buffer contents and resets the buffer.

CAs? -> CAs?00011000000010000010 (get version, does not reset any of the alarms in the buffer)

CAs! -> CAs!00011000000010000010 (set version, returns alarms, then resets the alarms buffer)

CAs! -> CAs!00001000000000000000 (same but now you see that you only get the alarm that is still active, the memorised alarms were reset by the previous get command).

And indeed, you can see that the 5th bit is still in alarm (meaning it is an active alarm).

If I now send las?4 (to get the description of the 5th bit, we start counting from 0), I can get the description of that bit.

las?4 -> las?ntcSeEqAllInterface;Interface

Now the command las? (Alarm index description - AIAAlarmDesc) returns a short textual description of the n-th alarm-buffer of the device.

The reply will consist of two parts separated by a semicolon ";".

The first part is the alarm name (intended for SW managing the device, like Newtec SEMS), while the second part is a short description of the alarm.

Now I did send following commands to clarify the las commands:

las?0 -> las?ntcSeEqAlResFlag;Device reset flag
las?1 -> las?ntcSeEqAlSelfTest;Self test
las?2 -> las?ntcSeEqAlIncompat;Incompatibility
las?3 -> las?ntcSeEqAlGenDev;General device
las?4 -> las?ntcSeEqAlInterface;Interface
las?5 -> las?ntcSeEqAlRefClock;Reference clock
las?6 -> las?ntcSeEqAlDevTemp;Device temperature
las?7 -> las?ntcSeEqAlPowSup;Power supply voltage
las?8 -> las?ntcSeEqAlMoInpFram;Input framing
las?9 -> las?ntcSeEqAlMoTbSync;Timebase sync.
las?10 -> las?ntcSeEqAlMoBufUfl;Buffer underflow
las?11 -> las?ntcSeEqAlMoBufOfi;Buffer overflow
las?12 -> las?ntcSeEqAlMoClkPll;Clock PLL
las?13 -> las?ntcSeEqAlMoSynth;Synthesiser
las?14 -> las?ntcSeEqAlMoRfPlo;RF phase lock DRO
las?15 -> las?ntcSeEqAlBissSum;BISS summary
las?16 -> las?ntcSeEqAlMcModule;Internal M&C module
las?17 -> las?ntcSeEqAlIntfModule;Interface module
las?18 -> las?ntcSeEqAlModModule;Internal modulator
las?19 -> las?ntcSeEqAlArchitecture;Device architecture

So in short, first send the CAs! command to get the current alarms and reset them.

Then send it again and you can see from the reply which alarms were memorised (those that are now 0) and which one are still active.

Then from the position in the bit string, you can determine which alarm it is.

Then you can act according to the severity you attribute to that alarm and determine your action to be taken.

The advantage of working with a dynamic alarm string is that whenever another device is connected to the management system that has more or less possible alarms (and which could hold another position on the alarm string), the management system will still act in the same way. Another device could have alarm bits at another position (or a different number) because of differences in hard- and software capabilities.

12 Questions and answers

Following is a list of typical questions we received from some of our customers. Please check this list to find out if the answer to your question hasn't been answered already. In case a question remains unanswered, feel free to contact our technical support (contact details can be found on <http://www.newtec.be/index.php?id=contactsupport>)

Q1: In v2, can we know the set of all possible alarms?

A: First send the command `Nas?` that will return the number of available alarms. Then perform a loop with `las?n` where $n=0$ to the number returned by `Nas?-1`. This will return the mnemonic and name of each available alarm (e.g. `las?14 -> las?ntcSeEqAlMoRfPlo;RF phase lock DRO`) with `ntcSeEqAlMoRfPlo` being the mnemonic of this command (if you drop the `ntcSeEq` of this string, you can search in the html RMCP usermanual for this command). The command also returns a textual description of the alarm.

Q2: In v2, the reply of “las“ command consists of two part, how can NMS recognize specific alarm? Can it be recognized by the first part of such reply? (Is this string unique and constant for each alarm?)

A: The command `las` (Identification of alarm) returns the mnemonic and name of the interrogated alarm (e.g. `las?14 -> las?ntcSeEqAlMoRfPlo;RF phase lock DRO`) with `ntcSeEqAlMoRfPlo` being the mnemonic of this command (if you drop the `ntcSeEq` of this string, you can search in the html RMCP usermanual for this command). The command also returns a textual description of the alarm.

Q3: Does parameter “Test function” in v1 (SMm) exist in v2? If yes, to which parameter in v2 should we refer to?

A: The parameter `SMm` has been redefined from “test functions” to “device mode”. (Device mode – `SyDevMode`) defines the device operating mode. There are 2 modes:

- Normal mode: This is the standard operating mode which enables the default set of parameters that are most frequently used.
- Expert mode: This mode gives an "expert" operator access to an additional set of more advanced parameters.

A password is required to switch to expert mode. This password is the model number of the device (e.g. 2180 for a NTC/2180 DVB L-band modulator) or the software identification number in case of a board (e.g. 6161 for a DVB-S modulator board). Commands that can only be changed in expert mode are indicated in this user manual by "access level: expert only".

To set the device mode via RMCP, the password had to be given as a structured command (e.g. to set the device mode to expert in a NTC/2277, the command `SMm!2,2277` will need to be send).

Q: 4. In v1 we have “Device capability” parameter (SLc), in v2 we have a number of different capabilities. What capability should we refer to in v2 in order to know the value of the same parameter as in v1? And for “Device identification” (0) from v1 and “Hardware description” (SSH) from v1.

A: Following parameters are available that uniquely define the device (see also ../Architecture/General)

- Device serial number - SyDevSn
- Device hardware identification - SyDevHwId
- Device hardware version - SyDevHwVer
- Device hardware capability - SyDevHwCapab
- Device software identification - SyDevSwId
- Device software version - SyDevSwVer
- Device software capability - SyDevSwCapab
- Device capability - SyDevCapab
- Product identification number – SyDevProdId

There is no longer a 1-to-1 mapping of the values of these parameters between the 20xx, 21xx or 22xx series of devices.

Q5: In v1 we have “Interface identification”, what parameter from v2 has the same meaning?

A: The same list of parameters as described above is available for each of the boards in the device. The same commands have to be used but with direct addressing. Only the addresses of the modulator and interface board are accessible for customers.

E.g.: SLs? will return the serial number of the general device, but proceeding the command by the direct address of the modulator board will return the serial number of the modulator board.

```
TXD-> ADR: 100, DATA: SLs?                // get general serial number
RXD-> ADR: 100, DATA: SLs?05031401        // reply
TXD-> ADR: 100, DATA: :-1:1:dSLs?         // get serial number of interface board
RXD-> ADR: 100, DATA: :-1:1:dSLs?05011177 // reply
TXD-> ADR: 100, DATA: :-1:1:tSLs?         // get serial number of modulator board
RXD-> ADR: 100, DATA: :-1:1:tSLs?05030901 // reply
```

The following hardware addresses are available:

- :-1:1:d for interface boards
- :-1:1:t for modulator boards
- :-1:1:e for RF converters

Warning! Direct addressing of other parameters (besides those in the architecture menu) is also possible but it is strongly advised not to use this to command the device. The top layer of the general monitor and control software controls the boards and should be the only controller of the boards. Using direct addressing of control parameters of boards may result in unwanted behaviour and erroneous transmissions!

Q6: Self-test: in v1 we have a number of commands to perform self test and get the result of it (SSs,Sss,SsS), from our understanding in v2 we have only one command to perform such test (GTS). How can NMS get the result after performing such test by v2?

A: The command “self test” does not exist anymore in v1.0 and v2.0 for the NTC/21xx and NTC/22xx series of equipment. The result of the boot self test is however included in the alarm string in location ‘0’. There is no possibility to trigger a self test during normal operation.

Q7: In v1 we have only one type of reset, in v2 we have number of possibilities to perform reset. What type of reset from v2 is equal to the reset from v1?

A: The reset command (Device reset – SyDevRst) has following enumerations possible:

- Soft reset (SRr!1) will send the reset command to all boards
- Factory reset (SRr!2) results in the deletion of all saved configurations (including the last active configuration) followed by a reboot to factory default settings.
- Hard reset (SRr!4) will power-cycle the device.
- Data-path reset (SRr!5) is used for hard resetting all sub-modules handling the data flowing through the device (i.e. all boards except the M&C related hardware). This can be useful to recover from corrupted firmware on such sub-modules without having to do a full hard reset of the complete device.
- The selection upgrade (SRr!6) is used whenever an upgrade through "bucket-files" is performed

The v1.0 protocol as used on NTC/20xx devices only knew the hard reset (SRr!4) command.

Q8: Is device reset flag supported in v2? (in v1 the command is Srs)

A: The device reset flag is no longer supported in v2.0

Q9: Is “Modulator output and monitoring I-band” parameter supported in v2? (in v1 the command is TTm). Does “MoOutputEnable” parameter have the same meaning?

Yes, the command TTm in v1.0 is the “Internal L-band transmit (Modulator output)” command as used in NTC/20xx devices. Since in the NTC/21xx and NTC/22xx series the modulator is a separate board, the command TTm has been redefined as “Modulator board transmit – MoOutputEnable”

Q10: Has “IF output level” in v1 (TLI) the same meaning as “MoOutputLevel” in v2 (OOL)?

Yes, the command OOL sets the operational output level of the modulator.

Q11: Is “IF output 1 transmit ” parameter supported in v2? (In v1 the command is TTt)

This parameter (used in the former NTC/2077) is replaced by the general parameter TTm “Modulator board transmit – MoOutputEnable”, the “IF output 2 transmit” parameter is not supported anymore in the NTC/21xx and NTC/22xx series since those modulators have only 1 IF output.

13 Some comparison notes between the NTC/21xx and NTC/22xx series

Q1: In the NTC/21xx series the command Transmit clock – MoTxClk (rmcp header: Tri) existed. In the NTC/22xx series the command “Transmit clock – MoTxClk” has been removed.

A: This command has been replaced by a command that groups the Transmit Clock and Transmit data in one. It has been replaced by the command Baseband processing mode – IfBBProc. The following table reflects the relationship between the enums of the commands:

Transmit clock: External	> Baseband processing: off
Transmit clock: Internal	> Baseband processing: stuffing (upward adaption with MPEG null-packets)
Transmit clock: Rata adapter	> Baseband processing: MPEG RA (rate adaption with MPEG null-packets)
Transmit clock: ---	> Baseband processing: BISS RA (BISS scrambling + rate adaption)

In DVB-S2 mode, stuffing or rate adaption is not needed anymore since the DVB-S2 mechanism already uses dummy PLFRAMES whenever the transmit buffer runs empty.

Q2: The command RF transmit – CvRfoutTx (rmcp header: TTt) is not implemented.

A: This command has been implemented only from ntc6241 v1.07 and higher, previously a RF block up converter was not implemented yet.

Q3: The command ASI switching mode - MoAsiSwitch (rmcp header: TIm) is not implemented.

A: This command has been implemented only from ntc6241 v1.07 and higher, pit was simply forgotten in previous releases/

Q4: All NCR related commands are not implemented in the NTC/22xx series

A: All NCR commands are currently not implemented yet in the 22xx series. The NCR commands are only used in modulators that are used in Newtec' DVB-RCS HUB stations. No other customers than Newtec use the NCR commands. These commands will only be implemented when Newtec will use the NCR function on the NTC/22xx series. Also here no other customers than Newtec will use these commands

Q5: The command Alarm trigger – AAlarmTrig (rmcp header: Pat) is not implemented in the NTC/22xx series

A: This command was removed already end 2003 in the NTC/21xx series and was replaced by Current alarm mode – AAlarmMode (rmcp header: Cam). This command not only allows forcing alarms but can also be used to mask alarms.

Q6: The command Transmit data – MoTxData (rmcp header: TDi) has different enums when comparing NTC/21xx and NTC/22xx

A: The same command is used in the 21xx and 22xx series but in the 22xx series, the PRBS sequence is generated in another location (and can be true PRBS or an incremental counter (selected by Test generator source type – MoTgSrcType), therefore it needed a new enum. External data has enum '0' for both commands, enum 1 for PRBS in the 21xx series and enum 5 for test generator for the 22xx series.