Nanometrics Command Formats

Reference Supplement

Nanometrics Command Formats

Reference Supplement

Nanometrics Inc.
Kanata, Ontario
Canada

Nanometrics Command Formats Reference Supplement

The information in this document has been carefully reviewed and is believed to be reliable and current for the following firmware versions: Comms controller 5.62, Trident 1.72, HRD 6.0 and later. Nanometrics, Inc. reserves the right to make changes at any time without notice to improve the reliability and function of the product.

The information in this document is provided to CTBTO under Call Off contract 00/20/6004 and is intended for internal use by CTBTO and its network operators only. Further distribution of this information requires the written consent of Nanometrics Inc.

Note: Packet formats for Private Data Stream data are covered in the current revision of public document 14602 “Nanometrics Data Formats”.

Nanometrics, Inc.
250 Herzberg Road
Kanata, Ontario, Canada K2K 2A1
Tel (613)592-6776
Fax (613)592-5929
Email info@nanometrics.ca

Part number 14607R3
Release date 2003-06-12

Contents

NMXP Outbound Commands

This document contains information about the data format for Nanometrics NMXP outbound commands, with a separate section describing key management command and response messages. The information in this document is supplementary to the Nanometrics Data Formats Reference Guide.

1.1 Overview of NMXP format

Data received on the serial port of the instrument are packetized in NMXP format, and then these packets are embedded in standard UDP packets. This data transmission format facilitates the transfer of data along with a wide variety of status information from an instrument to a central site. The data format requires that the instrument have an accurate time source (i.e. GPS) for time tagging the data prior to transmission.

The following objectives were used in designing the data format:

- Support retransmit of packets for error correction: It supports error free transmission of data using retransmission requests of bad packets.
- Simple to implement, even on small microprocessors
- Expandable: The status information in the data format is expandable. As new status information messages are created, they can be added to the data format without affecting the existing information.
- Programmable frequency for status information: Most of the status messages can be transmitted at a user defined frequency. This allows the user to tailor the ratio of data to status information. This is important on limited bandwidth or noisy transmission media.
- Efficient bandwidth usage

1.1.1 Description of NMX packets

All of the data are gathered into sequenced and time stamped packets. These packets start with a synchronization word plus an Oldest packet available word and finish with a CRC. The packets consist of 17 byte “bundles” of data. Each bundle is an independent collection of data. Each packet contains a time stamp bundle followed by n data bundles. In order to word align packets, an odd number of bundles is used.

The number of bundles in a packet is a programmable parameter, with a range of 1-255. This allows the packet size to be tailored to the data link; short packets should be used on noisy, error-prone links. Packets may be the same size for the entire network, or different on each branch (a branch is connected to one RM-4 port) of the network. All instruments on a given branch must use the same packet size. Short messages must be padded out to the packet size. Outgoing packets may have a different packet size.

Definitions:

- Outbound data: data that is being transmitted from the central recording site to the field stations
- Inbound data: data that is being transmitted from the field stations to the central recording site
- Channel: a channel is a unique stream of information (e.g., serial port 1)
    - an instrument may transmit one or more channels of information
- Packet: a packet is a uniquely identifiable collection of information that is transmitted, composed of data “bundles”
    - a packet contains information from only one channel
    - inbound packets contain data, status, or configuration information
    - outbound packets contain retransmit requests, or configurations
    - outbound packets do not have the Oldest packet word
- Bundle: each bundle is an independent collection of data (for example, time stamp information, status information, or data)
- All data is represented in the little endian format (Intel format)

1.1.2 Transport-specific wrapping

The packets described in this chapter are augmented with header bytes to facilitate transport over specific lower-level protocols.

Packets carried via serial communications are preceded by a synchronization word and followed by a 2-byte CRC, as follows:

2 bytes	synchronization word = 0xAABB
N bytes	packet payload
2 bytes	16-bit CRC

where the CRC is computed using the algorithm provided in Appendix B of the Nanometrics Data Formats Reference Guide. The synchronization word and the CRC are sent in little-endian byte order.

Packets carried via UDP or TCP are preceded by a 12-byte header containing the following information:

4 bytes	synchronization word = 0x7ABCDE0F
4 bytes	message type = 2 for all NMXP commands
4 bytes	message content length (packet length)

Note that these parameters are encoded in big-endian byte order.

1.2 Outbound packets

Outbound packets may be from 0 to 284 bytes in length, depending on the packet type. Instruments do not request retransmission of outbound packets. The following outbound packet types have been implemented:

ReTx Request by Sequence Number Range packet	2
HRD/Trident/Europa Calibration packet	6
Generic command packet (key management)	7

1.2.1 ReTx Request and HRD Calibration packets

ReTx Request and HRD/Trident/Europa Calibration packets are a fixed length of 26 bytes. The 20-bytes type-specific section contains the packet type, information header, and data sections. See sections 1.2.1.1 and 1.2.1.2 for descriptions of these types of outbound commands.

Basic format for ReTx Request and HRD/Trident/Europa Calibration packets:

2 bytes	Instrument ID (bits 0-10 serial number, bits 11-15 model type)
4 bytes	Packet time in seconds (UT)
1 byte	Packet type
3 bytes	Packet information header
4 x 4 bytes	Data section

1.2.1.1 Retransmission Request by Sequence Number Range packet

This packet is used to request retransmission of data packets from the instrument. The packets to be retransmitted are identified by the data type, channel, and sequence number. Currently retransmission request is supported for time series, SOH, Log, and transparent serial (including authentication) packets.

2 bytes	Instrument ID (bits 0-10 serial number, bits 11-15 model type)
4 bytes	Packet time in seconds (UT)
1 byte	Packet type = 2
1 byte	Channel
1 byte	Data type
1 byte	Spare
4 byte	First sequence number requested
4 bytes	Last sequence number requested
8 byte	Spare

Notes:

(1) Data type is the data packet type requested (1 = time series, 2 = SOH, 5 = log, 6 = transparent serial).

(2) If the data type byte is zero, type can be determined from channel (0-5 = time series, –1 = SOH).

(3) The data type byte was added to support new features for Libra (e.g., transparent serial). It is not used on HRDs.

(4) For time series, channel may be 0 to 5.

(5) For SOH, channel should be –1.

(6) For transparent serial, channel should be the corresponding serial port number.

(7) For authentication, channel should be 16 plus the corresponding time-series channel number.

(8) The packet time is the time at which the command is issued. It has no relationship to the data requested.

1.2.1.2 HRD/Trident/Europa Calibration packet

2 bytes		Instrument ID (bits 0-10 serial number, bits 11-15 model type)
4 bytes		Packet time in seconds (UT)
1 byte		Packet type = 6
1 byte		Channel: bit field indicating which channel relays to close
2 bytes		Spare
4 bytes	us-int32	Start time (long seconds)
4 bytes	float	Frequency in Hz (0.01 <= freq < sample rate/2)
4 bytes	float	Amplitude in volts (0 <= V <= 5.0) or amperes (0 <= A <= 0.035)
2 bytes	us-int16	Duration in seconds (0 < dur <= 3600)
1 byte		Control: Bit field specifying which digital control lines to be toggled
1 byte		Calibration type: 0 HRD voltage calibration 1 Trident voltage calibration 2 Trident current calibration

Notes:

(1) Digitiser does only one calibration at a time. If it receives a new cal packet while calibrating, it aborts the current calibration and sets up the new one.

(2) If the channel and control fields are both zero, the digitiser aborts the current calibration.

(3) Start time normally indicates start time in seconds since 1970.

The following special cases are supported for HRD:

StartTime = 1 start at beginning of next second
60 start at beginning of next minute
3600 start at beginning of next hour

For Trident, the only permitted value is:

StartTime = 1 start at beginning of next second

(4) If start time < current time, HRD discards the packet.

(5) If frequency, amplitude, or duration is out of range, digitiser discards the packet.

(6) Invalid packet still aborts previous calibration.

(7) The digitiser will not abort the current calibration when it receives a repeated calibration message with the same time, frequency, amplitude, duration, and type.

(8) For mass centring, set the channel field to zero and toggle the appropriate digital control lines.

(9) The packet time is the time at which the command is issued. It has no relationship to the data requested.

1.2.2 Generic command packet

Generic command packets are used for key management outbound commands. They are of variable length, from 17 to 284 bytes. See section 1.3 for descriptions of key management outbound commands and inbound responses.

Basic format for Generic command packet:

2 bytes	Instrument ID (bits 0-10 serial number, bits 11-15 model type)
4 bytes	Packet time in seconds (UT)
1 byte	Packet type = 7
1 byte	Command subtype
2 bytes	Number of bytes of payload data = N
N bytes	Payload, with format dependent on response type

1.3 Key management messages

Key management uses generic message types within NMXP format for the exchange, via IP or serial communications. Key management message payloads should always be in most significant byte-first order. Command packet payloads may be from 0 to 284 bytes in length. Key management follows a command-request protocol, in which each command packet elicits exactly one response packet.

1.3.1 Commands summary

	Command subtype	Payload (bytes)
generateKeyPair	1	284
getPublicKey	2	4
changeKeyID	3	12
getKeyList	4	0
setActiveKey	5	8
deleteKeyPair	6	8

1.3.2 Responses summary

	Response subtype	Payload (bytes)
Generic response confirmation	0	4
Generic error code	1	84
DSA key pair	2	448
Key list	3	24

1.3.2.1 Command response packet - Incoming packet

2 bytes	Instrument ID (bits 0-10 serial number, bits 11-15 model type)
4 bytes	Packet time in seconds (UT)
1 byte	Packet type = 8
4 bytes	Integer seconds since 1970, LSB first
2 bytes	Subseconds = 0
2 bytes	Instrument ID of source, LSB first
4 bytes	Sequence number (unused)
2 bytes	Number of bytes of payload data = N
1 byte	Response subtype
1 byte	Command subtype of corresponding command
N bytes	Payload, with format dependent on response type

1.3.3 Key management commands

1.3.3.1 generateKeyPair

The generateKeyPair command instructs the firmware to generate a new DSA key pair using specified parameters (p, q, g), and store it with a unique keyID. Parameters which are less than the field length provided must be right-justified, and padded on the left with zeroes.

4 bytes	Key ID for this key pair (if 0, the firmware will generate an ID)
128 bytes	P - byte array containing the DSA p parameter, MSB first
20 bytes	Q - byte array containing the DSA q parameter, MSB first
128 bytes	G - byte array containing the DSA g parameter, MSB first
4 bytes	CRC

1.3.3.2 getPublicKey

The getPublicKey command instructs the firmware to return a message containing the DSA key (along with its parameters) for the specified keyID.

4 byte	Key ID of requested key

1.3.3.3 changeKeyID

The changeKeyID command instructs the firmware to change the key ID used to identify a specified key pair. This allows IDC to change the keyID for a key pair after it has been generated (for example, to correspond to the serial number of a security certificate issued for the public key).

4 bytes	Existing key ID to be changed
4 bytes	New key ID
4 bytes	CRC

1.3.3.4 getKeyList

The getKeyList command instructs the firmware to return a message containing all keyIDs currently defined.

0 bytes

1.3.3.5 setActiveKey

The setActiveKey command instructs the firmware to use the specified key for data authentication.

4 bytes	Key ID for key pair to be used for authentication (must exist)
4 bytes	CRC

1.3.3.6 deleteKeyPair

The deleteKeyPair command instructs the firmware to delete the specified key pair from the token.

4 bytes	Key ID for the key pair to be deleted (may not delete the active key pair)
4 bytes	CRC

1.3.4 Key management responses

1.3.4.1 Generic response confirmation

The Generic response confirmation message is sent to confirm that a specified command has been carried out successfully. This message is sent in response to commands 3 (changeKeyID) and 6 (deleteKeyPair).

Note The message header contains the command subtype of the originating command.

4 bytes	Time stamp of command being acknowledged

1.3.4.2 Generic error response

The Generic error response message is sent to indicate that a specified command has not been carried out.This message can be sent in response to any of the commands.

Note The message header contains the command subtype of the originating command.

4 bytes	Time stamp of command being acknowledged
80 bytes	Zero-terminated error message

1.3.4.3 DSA key pair

The DSA key pair message is sent in response to command type 1 (generateKeyPair) or 2 (getPublicKey). Parameters which are less than the field length provided must be right-justified and padded on the left with zeroes.

4 bytes	Key ID for this key pair (if 0, the firmware will generate an ID)
128 bytes	P - byte array containing the DSA p parameter, MSB first
20 bytes	Q - byte array containing the DSA q parameter, MSB first
128 bytes	G - byte array containing the DSA g parameter, MSB first
128 bytes	G - byte array containing the DSA key (y), MSB first
40 bytes	DSA signature for payload

1.3.4.4 Key list

The Key list message is sent in response to command type 4 (getKeyList) or 5 (setActiveKey). This message provides a list of the key IDs available on the token and indicates which is active (currently used for signing).

4 bytes	Key ID of active key pair
4 bytes	Number of key pairs on token (maximum 4)
16 bytes	4 byte key IDs for each key pair on token (maximum 4)