NVISION Serial Protocol

General Messages

NP0009–02
Rev B

TABLE OF CONTENTS

1.0 Introduction

NVISION Serial Protocol refers to the data format of messaging used to control, configure, and otherwise communicate with NVISION products. The structure of this protocol is maintained throughout the NVISION product line. Note that NVISION Serial Protocol may also be referred to as "NV Serial Protocol" or "NVSP".

2.0 References

2.1 Specifications and Standards

- SMPTE 207M – Digital Control Interface – Electrical and Mechanical Characteristics

2.2 Other Protocol Documents

- NP0015 – NVISION Protocol – Shared Definitions

3.0 Document Conventions

- Decimal numbers are expressed with no radix.
- Hexadecimal numbers are prefixed with a radix of "0x" as in the "C" programming language.
- When 16 bit values are shown in message formats, they are indicated using two bytes, the most significant of them labeled "high byte" and the least significant of them labeled "low byte".
- When message data fields' positions are fixed, they are indicated with a numeral, such as "DATA 0". When those positions are variable they are designated "DATA".
- Data that may repeat are indicated with a curly brace ("}") and the designation "may repeat". Such data shall be repeated as a complete set.
- A data value of '??" is used in places where the value of the data is context dependant.

4.0 Physical and Data Link Layer

Device connections typically employ standard 9–pin female connectors using the RS422 (SMPTE 207M), RS485 or RS232 electrical characteristics. See the product manual for:

1) The pinout of the connectors on a specific product
2) The baud rates supported by the connectors of a specific product.

All characters use the same format:

1 Start bit
8 Data bits
1 Stop bit
No Parity bit

RS485 ports support multi–drop operation. All NVISION RS485 drivers tri–state their transmit lines when not otherwise driving them.

4.1 Transport Layer

All NVSP messages use the following format.

STX
DDID
DDAD
SSID
SSAD
COUNT
COMMAND
DATA
CHECKSUM

Where the header characters are defined as:

STX:	Start of transmission (0xFF)
DDID:	Destination device ID
DDAD:	Destination device address
SSID:	Source device ID
SSAD:	Source device address
COUNT:	Count of command and data characters, excluding the checksum
COMMAND:	Device specific command

The DDID/DDAD and SSID/SSAD are swapped in response messages and are interpreted in various ways depending on the addressing mode used.

The header is followed by zero or more DATA characters.

DATA: Command dependant, 0 to 240 characters

The message is terminated with a checksum.

CHECKSUM: Modulo 256 sum of all characters except the STX
(1 byte checksum, simple sum, carry ignored.)

Note: sample “C” code for computing the checksum is given in appendix 3.

4.2 Reserved SSID/SSAD values for Third Party Control System

The SSID value of 0xF0 is reserved for use by third party control systems. The SSID/SSAD combination of 0xF0, 0x00 is available for use by any third party control system. The table below shows some SSID/SSAD combinations reserved for us by specific systems.

SSID	SSAD	3rd Party Control System
0xF0	0x00	generic
0xF0	0x01	Miranda
0xF0	0x05	Harris Automation
0xF0	0x10	SONY

5.0 Addressing Modes

Global Addressing Mode is the addressing mode commonly used by 3rd party developers. There are other addressing modes discussed in appendix 4.

5.1 Global Addressing Mode

In Global Addressing mode (DDID = 0xFE), the DDAD specifies a router level as configured on individual NVISION router controllers or an address as configured on other devices. Responses to Global Addressing mode messages may alter the SSID field to indicate the Device ID of the device responding.

6.0 General Command Messages

6.1 Device Present Command 0x01

Requests the presence of a device assigned to a specific level.

Message Format:

STX	0xFF
DDID	0xFE = Global addressing mode
DDAD	Level
SSID	??
SSAD	??
COUNT	0x01
COMMAND	0x01
CHECKSUM	??

Responses: 0x04: ACK, Non–router device is present
0x09: Main server is present (NV9303, NV9304 and NV9370 only)
0xD9: Router Status, router device is present
0x80: Error response

6.2 Manufacturer and Product ID Command 0x10

The addressed device's Manufacturer and Product ID are requested.

Message format:

STX	0xFF
DDID	Any addressing mode
DDAD	??
SSID	??
SSAD	??
COUNT	0x01
COMMAND	0x10
CHECKSUM	??

Responses: 0x90: Manufacturer and Product ID Response
0x80: Error response

6.3 Software Version Command 0x11

The addressed device's software versions are requested.

Message format:

STX	0xFF
DDID	Any addressing mode
DDAD	??
SSID	??
SSAD	??
COUNT	0x01
COMMAND	0x11
CHECKSUM	??

Responses: 0x91: Software Version Response
0x80: Error response

6.4 Extended Software Version Command 0x15

The addressed controller's software versions are requested. If the controller is an older product that stores a classic NVISION style part number, this query will return an unsupported or unknown error and the query should be re-issued using the 0x11 Software Version Command.

Message format:

STX	0xFF
DDID	Any addressing mode
DDAD	??
SSID	??
SSAD	??
COUNT	0x02
COMMAND	0x15
DATA 0	Version data format 0x00 = Version format 0x01 = Version and Activity format
CHECKSUM	??

Responses: 0x95: Software Version Response
0x80: Error response

7.0 General Response Messages

7.1 Acknowledgement (ACK) Response 0x04

Acknowledgement indicates that a message has been received with valid format, character count and checksum, and that its command and data characters are valid and within range for the device receiving the message.

General Acknowledgement message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x01
COMMAND	0x04
CHECKSUM	??

Sequenced Acknowledgement message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x02
COMMAND	0x04
DATA 0	Sequence number
CHECKSUM	??

7.2 Error Response 0x80

An error is reported in response to a command.

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x02
COMMAND	0x80
DATA 0	Error codes: 0x00 = Undefined 0x01 = Invalid Data 0x02 = Device cannot execute command 0x03 = Checksum error in message 0x04 = Unknown command 0x05 = Communications port overrun error 0x06 = Count Error. The message byte count does not correspond to the expected count for the command(s) received. 0x07 = Unmapped error. A source or destination resolves to no physical source or destination. 0x08 = Parity error 0x09 = Command not supported by this device. 0x0A = Busy (typically the result of a queue overflow) 0x0B = Invalid addressing mode 0x0C = Timeout error
CHECKSUM	??

7.3 Error Response to Sequenced Message 0x81

An error is reported in response to a sequenced command.

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x03
COMMAND	0x81
DATA 0	Error Codes: Same as those for 0x80 response
DATA 1	Sequence number
CHECKSUM	??

7.4 Extended Status Response 0x82

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x03
COMMAND	0x82
DATA 0	High status
DATA 1	Low status
CHECKSUM	??

Notes:

1) Status values are the “Extended Status Values” defined NP0015.

7.5 ASCII Text Response Message 0x83

This response message contains ASCII text of up to 240 ASCII characters.

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	??
COMMAND	0x82
DATA	ASCII character
CHECKSUM	??

} may repeat

7.6 Unicode Text Response Message 0x84

This response message contains Unicode text of up to 120 Unicode characters.

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	??
COMMAND	0x83
DATA 0	High Unicode Character
DATA 1	Low Unicode Character
CHECKSUM	??

} may repeat

7.7 Manufacturer and Product ID Response 0x90

Response to a Manufacturer and Product ID command.

Message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	0x03
COMMAND	0x90
DATA 0	Manufacturer ID: 0x00 = NVISION 0xFF = unknown or not defined
DATA 1	Product ID (see DeviceIDs in NP0015-00)
CHECKSUM	??

7.8 Software Version Response 0x91

Response to a Software Version command. A device with multiple processors reports all processors that are responding to the query. A processor that is currently running its boot program shall report only its boot version, and a processor that is currently running its application program shall report both its boot and applications versions. The count is one greater than the number of versions reported times 7.

The software assembly number and version data are derived from the NVISION "SV" part number. For example, a part number of SV1234-567890 would result in:

- DATA 2 equal to 0x04 and DATA 3 equal to 0xD2 (1234 decimal is equal to 0x04D2)
- DATA 4 equal to 0x38 (56 decimal is equal to 0x38)
- DATA 5 equal to 0x4E (78 decimal is equal to 0x4E)
- DATA 6 equal to 0x5A (90 decimal is equal to 0x5A)

Other manufacturers are free to define how they choose to format these fields.

Message format:

STX	0xFF	
DDID	??	
DDAD	??	
SSID	??	
SSAD	??	
COUNT	See above	
COMMAND	0x91	
DATA 0	Firmware ID (see Appendix 1 for definitions)	may repeat
DATA 1	Firmware Type (see Appendix 1 for definitions)	
DATA 2	Software assembly number (high byte)	
DATA 3	Software assembly number (low byte)	
DATA 4	Version MSB	
DATA 5	Version MID	
DATA 6	Version LSB	
CHECKSUM	??	

7.9 Extended Software Version Response 0x95

Response to an Extended Software Version command. A device with multiple processors reports all processors that are responding to the query. A processor that is currently running its boot program shall report only its boot version, and a processor that is currently running its application program shall report both its boot and applications versions.

The response is broken into multiple messages when the "number of messages to follow" field is greater than 0. Each subsequent response will have a "number of messages to follow" field that is one less than the previous response until it reaches zero. For example if 5 responses are required the first will have a number of messages to follow value of 4 meaning four more responses follow. The last response will have a number of messages to follow value of 0.

Version data message format (message format field = 0x00).

The message count field is variable and depends on the length of the descriptive text field.

The part number consists of as many as 16 ASCII part number characters. The part number revision consists of as many as four ASCII revision characters. When the ASCII data do not completely fill their fields, characters are supplied beginning from the lowest ordered DATA field, terminated with at least one null (0x00), and the remaining characters in the field become don't cares. For example, the fields of a two-character part number revision of "A1" are transmitted in the following order:

0x41 ('A'), 0x31 ('1'), 0x00 (NULL), 0x?? (don'tcare)

The descriptive text field consists of a string of zero or more ASCII characters terminated with the NULL (0x00) character. The length of this string is limited only by the maximum message size. Another set of version data may begin immediately following the NULL that terminates this string.

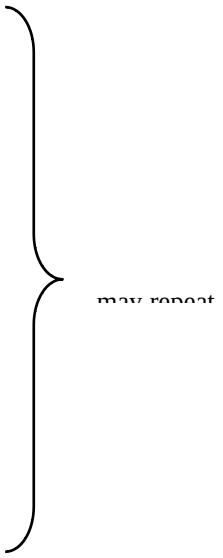
Version and activity data message format (message format field = 0x01).

> The message is identical to the format above, with the addition of a single byte field that represents whether or not an application is executing.

No other message formats are supported.

Version data message format:

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	See above
COMMAND	0x95
DATA 0	Number of messages to follow
DATA 1	0x00 = Version data format
DATA 2	Firmware ID
DATA 3	Firmware Type (see Appendix 1 for definitions)
DATA 4	First Part Number Character (ASCII)
DATA 5-18	Intermediate Part Number Characters (ASCII)
DATA 19	Final Part Number Character (ASCII)
DATA 20	First Part Number Revision Character (ASCII)
DATA 21-22	Intermediate Part Number Revision Characters (ASCII)
DATA 23	Final Part Number Revision Character (ASCII)
DATA 24	Major revision (high byte)
DATA 25	Major revision (low byte)
DATA 26	Minor revision (high byte)
DATA 27	Minor revision (low byte)
DATA 28	Patch level (high byte)
DATA 29	Patch level (low byte)
DATA 30	Build number (high byte)
DATA 31	Build number (low byte)
DATA 32-??	Descriptive text (ASCII)
CHECKSUM	??



Version and activity data message format

STX	0xFF
DDID	??
DDAD	??
SSID	??
SSAD	??
COUNT	See above
COMMAND	0x95
DATA 0	Number of messages to follow
DATA 1	0x01 = Version and activity data message format
DATA 2	Firmware ID
DATA 3	Firmware Type (see Appendix 1 for definitions)
DATA 4	Active Flag, 0x00 = idle 0x01 = executing
DATA 5	First Part Number Character (ASCII)
DATA 6-19	Intermediate Part Number Characters (ASCII)
DATA 20	Final Part Number Character (ASCII)
DATA 21	First Part Number Revision Character (ASCII)
DATA 22-23	Intermediate Part Number Revision Characters (ASCII)
DATA 24	Final Part Number Revision Character (ASCII)
DATA 25	Major revision (high byte)
DATA 26	Major revision (low byte)
DATA 27	Minor revision (high byte)
DATA 28	Minor revision (low byte)
DATA 29	Patch level (high byte)
DATA 30	Patch level (low byte)
DATA 31	Build number (high byte)
DATA 32	Build number (low byte)
DATA 33-??	Descriptive text (ASCII)
CHECKSUM	??

8.0 Reserved Command and Response Messages

The following commands may change format depending on the specific device. See the device specific protocol document to determine support and format of this command. In general, if this command is not mentioned in the device specific protocol document, then that device does not support this command.

0x06: Set Device State Command
0x12: System Status Command
0x92: System Status Response

The following command and response values are only used by routers (see NP0010-02 for details):

0x02: Timestamp Command
0x0B: Timestamp Response

The following command and response values are reserved for internal use only:

0x03: Real Time Clock Command
0x07: Message Carrier Command
0x08: Command
0x0C: Real Time Clock Response
0x13: Report Controller Type Command
0x14: Card Status Command
0x87: Message Carrier Response
0x88: Response
0x93: Report Controller Type Response
0x94: Card Status Response

The following command and response values are no longer used:

0x05: Negative Acknowledgement (NAK) Response
0x09: Response Main Server Present Response
0x71: Log Message
0x7E: ASCII Message
0x8F: ASCII Error Message

9.0 Revision History

9.1 Revision A

- Released: April 30, 2002

9.2 Revision B

- Released: June 22, 2004
- Removed the following commands and responses from this document:
    1. 0x03: Real Time Clock Command
    2. 0x05: Negative Acknowledgement (NAK) Response
    3. 0x06: Set Device State Command
    4. 0x09: Response Main Server Present Response
    5. 0x0B: Timestamp Response
    6. 0x0C: Real Time Clock Response
    7. 0x12: System Status Command
    8. 0x13: Report Controller Type Command
    9. 0x71: Log Message
    10. 0x7E: ASCII Message
    11. 0x8F: ASCII Error Message
    12. 0x92: System Status Response
    13. 0x93: Report Controller Type Response

- Added the following responses:
    1. 0x82: Extended Status Response
    2. 0x83: ASCII Text Response Message
    3. 0x84: Unicode Text Response Message

- Added section 4.2, Reserved SSID/SSAD values for Third Party Control System

- Moved description of all addressing modes except Global Addressing Mode from section 5 to Appendix 3.

- Removed Appendix 2, Buss ID definitions.

- General cleanup of the document to make it easier to use by 3rd party developers.

Appendix 1: Firmware ID and Firmware Type definitions

Firmware ID definitions

Some messages in this protocol use these firmware identifiers to specify the messages' target.

0x00 through 0x0F = APP0 through APP15 (Application data), respectively, or Boot0 through Boot15, respectively.
0x10 through 0x2F = PLD0 through PLD31 (Programmable Logic Device data), respectively.
0x30 through 0x3F= IF0 through IF15 (Intermediate File data), respectively

Products with a single application will use APP ID 0x00.

Firmware Type definitions

In general, if a product has only one software version, then the Software ID is always 0x00.

For NVISION routers the Firmware Types are:

0x00 = Boot, PLD
0x01 = Application

Firmware ID and Firmware Type combinations

The EM0134, EM0209, and EM3510 router controllers utilize three processors and use the following combinations of Firmware ID and Type:

Firmware ID	Firmware Type	Firmware description
0x00	0x00	Diagnostic (DIAG) Processor Boot
0x01	0x00	Matrix (or MTRX) Processor Boot
0x02	0x00	Communications (or COMM) Processor Boot
0x00	0x01	Diagnostic (DIAG) Processor Application
0x01	0x01	Matrix (or MTRX) Processor Application
0x02	0x01	Communications (or COMM) Processor Application
0x10	0x00	PLD0 – used with EM0209 only

Appendix 2: Example "C" code for computing checksum

```
/*****************************************************************************
;
;       NVISION PROTOCOL MESSAGE OFFSETS
;
;*****************************************************************************/

enum    NV_MSG_OFFSETS
        {
        NV_STX_OFFSET,              /* offset to STX field */
        NV_DDID_OFFSET,             /* offset to Destination Device ID header field */
        NV_DDAD_OFFSET,             /* offset to Destination Device ADdress header field */
        NV_SSID_OFFSET,             /* offset to Source device ID header field */
        NV_SSAD_OFFSET,             /* offset to Source device ADdress header field */
        NV_COUNT_OFFSET,    /* offset to count of command and data characters field */
        NV_CMD_OFFSET,              /* offset to command field */
        NV_DATA_OFFSET              /* offset to start of data field(s) */
        };

#define  NV_STX_CHAR 0xFF     /* start of transmission character */

/*****************************************************************************
;
;       Name: ComputeNvMsgCksum
;
;       Description: returns the computed sum of a message
;
;           Inputs: pointer to message
;
;          Returns: checksum
;
;*****************************************************************************/
unsigned char     ComputeNvMsgCksum(unsigned char* msg_ptr)
{
        unsigned char     accum = 0;
        unsigned char     index;
        unsigned char     count;

        /* store the message count */
        count = msg_ptr[NV_COUNT_OFFSET];

        /* omit NV_STX_CHAR */
        msg_ptr++;

        /* sum the message */
        for(index = 0 ; index < (count + NV_COUNT_OFFSET) ; index++)
        {
                accum += *msg_ptr++;
        }

        /* return message checksum */
        return(accum);
}
```

Appendix 3: Addressing modes not commonly used

Global Addressing Mode (see section 5.0), is the addressing mode most commonly used by 3rd party developers.

Only use the addressing modes below if Global Addressing mode will not work for you.

1) Device ID Addressing Mode

In Device ID addressing mode, the DDID specifies a device ID that is a function of that device's NVISION model designation. Device ID addressing mode messages intended for mixers, delays and other non–router devices use the DDAD to specify a card address, as configured on individual cards.

NVISION device ID's are defined in the document NP0015-00.

Device ID addressing mode messages intended for routers use the DDAD to specify a level, as configured on individual router controllers.

A source address (SSAD) of 0x00 is defined as the MASTER ADDRESS. A device using the master address is permitted to release locks and protects that normally require the address of the device that invoked that locks and protects.

2) Card Address Addressing Mode

In Card Address addressing mode (DDID = 0xFB), the DDAD specifies a configurable card address in a multi–card system. This addressing mode allows communication with a specific card whose address is known regardless of that card's other characteristics.

3) Direct Addressing Mode

Direct Addressing mode (DDID = DDAD = 0xFC) is used to directly address diagnostic ports. This addressing mode supports downloading application code and configuration data to device without knowing its specific address.

4) Volunteer Addressing Mode

Volunteer Addressing mode (DDID = DDAD = 0xFD) is used when a device transmits unsolicited log or error messages via a diagnostic port.