Innovative Technology

INTELLIGENCE IN VALIDATION

Protocol Manual

SSP

version GA138_2_2_2132A

Contents

Descriptions

Command/Event Tables

Commands

Events

<< back to index

Introduction

This manual describes the operation of the Smiley ® Secure Protocol **SSP**.

ITL recommend that you study this manual as there are many new features permitting new uses and more secure applications.
If you do not understand any part of this manual please contact the ITL for assistance. In this way we may continue to improve our product.
Alternatively visit our web site at www.innovative-technology.co.uk

Enhancements of SSP can be requested by contacting:
support@innovative-technology.co.uk

MAIN HEADQUARTERS
Innovative Technology Ltd
Derker Street, Oldham, England. OL1 4EQ

Tel: +44 161 626 9999 Fax: +44 161 620 2090
E-mail: support@innovative-technology.co.uk

Web site: www.innovative-technology.co.uk

Smiley ® and the **ITL Logo** are international registered trademarks and they are the property of **Innovative Technology Limited**.
Innovative Technology has a number of European and International Patents and Patents Pending protecting this product. If you require further details please contact ITL ®.

Innovative Technology is not responsible for any loss, harm, or damage caused by the installation and use of this product.

This does not affect your local statutory rights.

If in doubt please contact innovative technology for details of any changes.

<< back to index

General Description

Smiley ® Secure Protocol (SSP) is a secure interface specifically designed by ITL ® to address the problems experienced by cash handling systems in gaming machines.Problems such as acceptor swapping, reprogramming acceptors and line tapping areall addressed.

The interface uses a master-slave model, the host machine is the master and the peripherals (note acceptor, coin acceptor or coin hopper) are the slaves.

Data transfer is over a multi-drop bus using clock asynchronous serial transmissionwith simple open collector drivers. The integrity of data transfers is ensured through the use of 16 bit CRC checksums on all packets.

Each SSP device of a particular type has a unique serial number; this number is used to validate each device in the direction of credit transfer before transactions can takeplace. It is recommended that the encryption system be used to prevent fraud through busmonitoring and tapping. This is compulsory for all payout devices.

Commands are currently provided for coin acceptors, note acceptors and coinhoppers. All current features of these devices are supported.

FEATURES:

- Serial control of Note / Coin Validators and Hoppers
- 4 wire (Tx, Rx, +V, Gnd) system
- Open collector driver, similar to RS232
- High Speed 9600 Baud Rate
- 16 bit CRC error checking
- Data Transfer Mode
- Encryption key negotiation
- 128 Bit AES Encrypted Mode

BENEFITS:

- Proven in the field
- Simple and low cost interfacing of transaction peripherals.
- High security control of payout peripherals.
- Defence against surrogate validator fraud.
- Straightforward integration into host machines.
- Remote programming of transaction peripherals
- Open standard for universal use.

To help in the software implementation of the SSP, ITL can provide, C/C++ Code, C#.Net Code, DLL controls available on request. Please contact: support@innovative-technology.co.uk

<< back to index

Hardware layer

Communication is by character transmission based on standard 8-bit asynchronous data transfer.

Only four wires are required TxD, RxD, +V and ground. The transmit line of the host is open collector, the receive line of each peripheral has a 10Kohm pull-up to 5 volts.The transmit output of each slave is open collector, the receive input of the host has asingle 3k3 ohm pull-up to 5 volts.

The data format is as follows:

Encoding	**NRZ**
Baud Rate	**9600**
Duplex	**Full**
Start bits	**1**
Data Bits	**8**
Parity	**none**
Stop bits	**2**

Caution: **Power to peripheral devices would normally be via the serial bus. However devices that require a high current supply in excess of 1.5 Amps, e.g. hoppers, would be expected to be supplied via a separate connector.**

<< back to index

Transport Layer

Data and commands are transported between the host and the slave(s) using a packet format as shown below:

STX	SEQ/SLAVE ID	LENGTH	DATA	CRCL	CRCH

STX	Single byte indicating the start of a message - 0x7F hex
SEQ/ Slave ID	Bit 7 is the sequence flag of the packet, bits 6-0 represent the address of the slave the packet is intended for, the highest allowable slave ID is 0x7D
LENGTH	The length of the data included in the packet - this does not include STX, the CRC or the slave ID
DATA	Commands and data to be transferred
CRCL, CRCH	Low and high byte of a forward CRC-16 algorithm using the Polynomial (X16 + X15 + X2 +1) calculated on all bytes, except STX. It is initialised using the seed 0xFFFF. The CRC is calculated before byte stuffing.

PACKET SEQUENCING

Byte stuffing is used to encode any STX bytes that are included in the data to be transmitted. If 0x7F (STX) appears in the data to be transmitted then it should be replaced by 0x7F, 0x7F.
Byte stuffing is done after the CRC is calculated, the CRC its self can be byte stuffed. The maximum length of data is 0xFF bytes.
The sequence flag is used to allow the slave to determine whether a packet is a re-transmission due to its last reply being lost. Each time the master sends a new packet to a slave it alternates the sequence flag. If a slave receives a packet with the same sequence flag as the last one, it does not execute the command but simply repeats it's last reply. In a reply packet the address and sequence flag match the command packet.
This ensures that no other slaves interpret the reply as a command and informs the master that the correct slave replied. After the master has sent a command to one of the slaves, it will wait for 1 second fora reply. After that, it will assume the slave did not receive the command intact so it will re-transmit it with the same sequence flag. The host should also record the fact that a gap in transmission has occurred and prepare to poll the slave for its serial number identity following the current message. In this way, the replacement of the hosts validator by a fraudulent unit can be detected.
The frequency of polling should be selected to minimise the possibility of swapping a validator between polls. If the slave has not received the original transmission, it will see the re-transmission as a new command so it will execute it and reply. If the slave had seen the original command but its reply had been corrupted then the slave will ignore the command but repeat its reply. After twenty retries, the master will assume that the slave has crashed. A slave has no time-out or retry limit. If it receives a lone sync byte part way through receiving a packet it will discard the packet received so far and treat the next byte as an address byte.

<< back to index

Encryption Layer

PACKET FORMAT

Encryption is mandatory for all payout devices and optional for pay in devices. Encrypted data and commands are transported between the host and the slave(s) using the transport mechanism described above, the encrypted information is stored in the data field in the format shown below:

STX	SEQ/SLAVE ID	LENGTH	**DATA**	CRCL	CRCH

DATA

STEX	Encrypted Data

Encrypted Data

eLENGTH	eCOUNT	eDATA	ePACKING	eCRCL	eCRCH

STEX	Single byte indicating the start of an encrypted data block - 0x7E
eLENGTH	The length of the data included in the packet - this does not include STEX, COUNT, the packing or the CRC
eCOUNT	A four byte unsigned integer. This is a sequence count of encrypted packets, it is incremented each time a packet is encrypted and sent, and each time an encrypted packet is received and decrypted.
eDATA	Commands or data to be transferred
ePACKING	Random data to make the length of the length +count + data + packing + CRCL + CRCH to be a multiple of 16 bytes
eCRCL/eCRCH	Low and high byte of a forward CRC-16 algorithm using the polynomial (X16 + X15 + X2 +1) calculated on all bytes except STEX. It is initialised using the seed 0xFFFF

After power up and reset the slave will stay disabled and will respond to all commands with the generic response KEY_NOT_SET (0xFA), without executing the command, until the key has been negotiated. There are two classes of command and response, general commands and commands involved in credit transfer.

General commands may be sent with or without using the encryption layer. The slave will reply using the same method, unless the response contains credit information, in this case the reply will always be encrypted. Credit transfer commands, a hopper payout for example, will only be accepted by the slave if received encrypted. Commands that must be encrypted on an encryption-enabled product are indicated on the command descriptions for each command. The STEX byte is used to determine the packet type. Ideally all communications will be encrypted.

After the data has been decrypted the CRC algorithm is performed on all bytes including the CRC. The result of this calculation will be zero if the data has been decrypted with the correct key. If the result of this calculation is non-zero then the peripheral should assume that the host did not encrypt the data (transmission errors are detected by the transport layer). The slave should go out of service until it is reset.

The packets are sequenced using the sequence count; this is reset to 0 after a power cycle and each time the encryption keys are successfully negotiated. The count is incremented by the host and slave each time they successfully encrypt and transmit a packet. After a packet is successfully decrypted the COUNT in the packet should be compared with the internal COUNT, if they do not match then the packet is discarded.

<< back to index

Encryption Keys

The encryption key length is 128 bits. However this is divided into two parts. The lower 64 bits are fixed and specified by the machine manufacturer, this allows the manufacturer control which devices are used in their machines.
The higher 64 bits are securely negotiated by the slave and host at power up, this ensures each machine and each session are using different keys. The key is negotiated by the Diffie-Hellman key exchange method.
See: en.wikipedia.org/wiki/Diffie-Hellman

The exchange method is summarised in the table below. C code for the exchange algorithm is available from ITL.

Step	Host	Slave
1	Generate prime number GENERATOR	
2	Use command Set Generator to send to slave Check GENERATOR is prime and store	Check GENERATOR is prime and store
3	Generate prime number MODULUS	
4	Use command Set Modulus to send to slave Check MODULUS is prime and store	Check MODULUS is prime and store
5	Generate Random Number HOST_RND	
6	Calculate HostInterKey: = GENERATOR ^ HOST_RND mod MODULUS	
7	Use command Request Key Exchange to send to slave.	Generate Random Number SLAVE_RND
8		Calculate SlaveInterKey: = GENERATOR ^ SLAVE_RND mod MODULUS
9		Send to host as reply to Request Key Exchange
10	Calculate Key: = SlaveInterKey ^ HOST_RND mod MODULUS	Calculate Key: = HostInterKey ^ SLAVE_RND mod MODULUS

Note: ^ represents to the power of

<< back to index

Generic Commands and Responses

All devices must respond to a list of so-called Generic Commands as show in the table below.

Command	Code
Reset	0x01
Host Protocol Version	0x06
Get Serial Number	0x0C
Sync	0x11
Disable	0x09
Enable	0x0A
Get Firmware Version	0x20
Get Dataset Version	0x21

A device will respond to all commands with the first data byte as one of the Generic responses list below..

Generic Response	Code	Description
OK	0xF0	Returned when a command from the host is understood and has been, or is in the process of, being executed.
COMMAND NOT KNOWN	0xF2	Returned when an invalid command is received by a peripheral.
WRONG No PARAMETERS	0xF3	A command was received by a peripheral, but an incorrect number of parameters were received.
PARAMETERS	0xF4	One of the parameters sent with a command is out of range.
COMMAND CANNOT BE PROCESSED	0xF5	A command sent could not be processed at that time. E.g. sending a dispense command before the last dispense operation has completed.
SOFTWARE ERROR	0xF6	Reported for errors in the execution of software e.g. Divide by zero. This may also be reported if there is a problem resulting from a failed remote firmware upgrade, in this case the firmware upgrade should be redone.
FAIL	0xF8	Command failure
KEY NOT SET	0xFA	The slave is in encrypted communication mode but the encryption keys have not been negotiated.

<< back to index

Protocol Versions

An SSP Poll command returns a list of events and data that have occurred in the device since the last poll.

The host machine then reads this event list taking note of the data length (if any) of each event.

On order to introduce new events, SSP uses a system of **Protocol Version** levels to identify the event types and sizes a machine can expect to see in reponse to a poll. If this were not done, new unknown events with unknown datasize to a machine not set-up for these would cause the event reading to fail.

A host system should take note of the protocol version of the device connected and ensure that it is not set for a higer version that the one it is expecting to use.

The host can also check that the device can also be set to the higher protocol level, ensuring that expected events will be seen.

The listed events in this manual show the protocol version level of each event.

As part of the start-up procedure, the host should read the current protocol level of the device (using the set-up request command).

<< back to index

Banknote Validator

A Banknote Validator is a device which will scan, validate and stack a banknote it detects as valid or reject it from the front if not valid. Some banknote validators can be transformed into payout devices by the addition of a pay-out unit. All ITL™ Banknote validators support the SSP protocol described here.

The Banknote Validators have a default SSP Address of 0.

The setup request reponse table for banknote validator types:

Protocol versions less than 6:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x00 = Banknote validator
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +(n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device

Protocol versions greater than or equal to 6:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0 = Banknote validator
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +(n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device
Expanded channel country code	16 + (n * 2)	n * 3	Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are given only on protocol versions >= 6.
Expanded channel value	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.

<< back to index

Reject Codes

The banknote validator specification includes a command Last Reject Code.

Use this command after a note has been rejected to return a one-byte code to determine the cause of the note reject.

Table showing some reject codes (other codes may be used for future validation failures):

0x00	0	NOTE ACCEPTED	The banknote has been accepted. No reject has occured.
0x01	1	LENGTH FAIL	A validation fail: The banknote has been read but it's length registers over the max length parameter.
0x02	2	AVERAGE FAIL	Internal validation failure - banknote not recognised.
0x03	3	COASTLINE FAIL	Internal validation failure - banknote not recognised.
0x04	4	GRAPH FAIL	Internal validation failure - banknote not recognised.
0x05	5	BURIED FAIL	Internal validation failure - banknote not recognised.
0x06	6	CHANNEL INHIBIT	This banknote has been inhibited for acceptance in the dataset configuration.
0x07	7	SECOND NOTE DETECTED	A second banknote was inserted into the validator while the first one was still being transported through the banknote path.
0x08	8	REJECT BY HOST	The host system issues a Reject command when this banknote was held in escrow.
0x09	9	CROSS CHANNEL DETECTED	This bank note was identified as exisiting in two or more seperate channel definitions in the dataset.
0x0A	10	REAR SENSOR ERROR	An inconsistency in a position sensor detection was seen
0x0B	11	NOTE TOO LONG	The banknote failed dataset length checks.
0x0C	12	DISABLED BY HOST	The bank note was validated on a channel that has been inhibited for acceptance by the host system.
0x0D	13	SLOW MECH	The internal mechanism was detected as moving too slowly for correct validation.
0x0E	14	STRIM ATTEMPT	An attempt to fraud the system was detected.
0x0F	15	FRAUD CHANNEL	Obselete response.
0x10	16	NO NOTES DETECTED	A banknote detection was initiated but no banknotes were seen at the validation section.
0x11	17	PEAK DETECT FAIL	Internal validation fail. Banknote not recognised.
0x12	18	TWISTED NOTE REJECT	Internal validation fail. Banknote not recognised.
0x13	19	ESCROW TIME-OUT	A banknote held in escrow was rejected due to the host not communicating within the time-out period. The default timeout period is the same as the poll timeout i.e. 10 seconds.
0x14	20	BAR CODE SCAN FAIL	Internal validation fail. Banknote not recognised.
0x15	21	NO CAM ACTIVATE	A banknote did not reach the internal note path for validation during transport.
0x16	22	SLOT FAIL 1	Internal validation fail. Banknote not recognised.
0x17	23	SLOT FAIL 2	Internal validation fail. Banknote not recognised.
0x18	24	LENS OVERSAMPLE	The banknote was transported faster than the system could sample the note.
0x19	25	WIDTH DETECTION FAIL	The banknote failed a measurement test.
0x1A	26	SHORT NOTE DETECT	The banknote measured length fell outside of the validation parameter for minimum length.
0x1B	27	PAYOUT NOTE	The reject code cammand was issued after a note was payed out using a note payout device.
0x1C	28	DOUBLE NOTE DETECTED	More than one banknote was detected as overlayed during note entry.
0x1D	29	UNABLE TO STACK	The bill was unable to reach it's correct stacking position during transport.

0x1F	31	Credit card Detected	Devices applicable: NV9 Family tree

<< back to index

NV200 Command Table

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Reject	0x08	8
Last Reject Code	0x17	23
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Get Barcode Data	0x27	39
Configure Bezel	0x54	84
Poll With Ack	0x56	86
Event Ack	0x57	87
Get Counters	0x58	88
Reset Counters	0x59	89
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Get Build Revision	0x4F	79
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Ssp Download Data Packet	0x74	116
Hold	0x18	24
Setup Request	0x05	5

NV200 Event Table

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Barcode Ticket Validated	0xE5	229
Barcode Ticket Ack	0xD1	209
Note Path Open	0xE0	224
Channel Disable	0xB5	181
Initialising	0xB6	182

<< back to index

Command	Code hex	Code decimal
Sync	0x11	17

Implemented on	Encryption Required
NV200	**optional**

Description

SSP uses a system of sequence bits to ensure that packets have been received by the slave and the reply received by the host. If the slave receives the same sequence bit as the previous command packet then this is signal to re-transmit the last reply.

A mechanism is required to initially set the host and slave to the same sequence bits and this is done by the use of the SYNC command.

A Sync command resets the seq bit of the packet so that the slave device expects the next seq bit to be 0. The host then sets its next seq bit to 0 and the seq sequence is synchronised.

The SYNC command should be the first command sent to the slave during a session.

Packet examples

Send Sync command (0x11) with no data parameters and an address of "0", ensuring the next command starts with seq bit set to 0.

Host transmit:	**7F**	**80**	**01**	**11**	**65**	**82**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Reset	0x01	1

Implemented on	Encryption Required
NV200	**optional**

Description

Performs a software and hardware reset of the device.

After this command has been acknowledged with **OK (0xF0)**, any encryption, baud rate changes, etc will be reset to default settings.

Packet examples

No data parameters, sequence bit set and address 0

Host transmit:	**7F**	**80**	**01**	**01**	**06**	**02**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Host Protocol Version	0x06	6

Implemented on	Encryption Required
NV200	**optional**

Description

ITL SSP devices use a system of protocol levels to control the event responses to polls to ensure that changes would not affect systems with finite state machines unable to test for new events with non-defined data lengths.

Use this command to allow the host to set which protocol version to operate the slave device.

If the device supports the requested protocol **OK (0xF0)** will be returned. If not then **FAIL (0xF8)** will be returned

Packet examples

The slave supports the protocol version 8

```
Host transmit:   7F  80  02  06  08  03  94
 Slave Reply:    7F  80  01  F0  23  80
```

Host protocol version 9 not supported

```
Host transmit:   7F  80  02  06  09  06  14
 Slave Reply:    7F  80  01  F8  10  00
```

<< back to index

Command	Code hex	Code decimal
Poll	0x07	7

Implemented on	Encryption Required
NV200	**optional**

Description

This command returns a list of events occured in the device since the last poll was sent.

The SSP devices share some common events and have some unique events of their own. See event tables for details for a specific device.

A single response can contain multiple events. The first event to have occured will be at the start of the packet.

Packet examples

Poll command returning device reset and disabled response

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  F1  E8  BF  8C
```

Event response note credit channel 1 and note stacked

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  04  F0  EE  01  EB  B9  48
```

<< back to index

Command	Code hex	Code decimal
Get Serial Number	0x0C	12

Implemented on	Encryption Required
NV200	**optional**

Description

This command returns a 4-byte big endian array representing the unique factory programmed serial number of the device.

An optional data byte can be sent to request the serial number of attached devices. Setting the optional byte to 0 is the same as sending no optional byte.

NVR-280 (NV12):

1. Printer serial number

Note Float (NV11):

1. Notefloat serial number

Multi-Note Float (NV22):

1. Multi Note Float serial number

Smart System:

1. Smart System Feeder serial number

NV200:

1. Smart Payout / Smart Ticket serial number.
2. TEBS serial number.
3. Bunch Note Feeder serial number.

With NV4000:

0x11: Recycler 1 module

0x12: Recycler 2 module

0x13: Recycler 3 module

0x14: Recycler 4 module

0x15: Interface module

Packet examples

The device responds with 4 bytes of serial number data. In this case, the serial number is 01873452 = 0x1c962c. The return array is formatted as big endian (MSB first).

Host transmit:	**7F**	**80**	**01**	**0C**	**2B**	**82**				
Slave Reply:	**7F**	**80**	**05**	**F0**	**00**	**1C**	**96**	**2C**	**D4**	**97**

Optional byte to get payout serial number. The serial number is 01873452 = 0x1c962c. The return array is formatted as big endian (MSB first).

```
Host transmit:   7F  80  02  0C  01  35  A8
 Slave Reply:    7F  80  05  F0  00  1C  96  2C  D4  97
```

<< back to index

Command	Code hex	Code decimal
Disable	0x09	9

Implemented on	Encryption Required
NV200	**optional**

Description

Disabled the slave device from operation.

For example, this command would block a banknote validator from allowing any more banknotes to be entered.

For most SSP devices, the default state is to be disabled after reset.

Packet examples

Single byte command with no parameters

Host transmit:	**7F**	**80**	**01**	**09**	**35**	**82**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

NV11 when note float is jammed/disconnected responds COMMAND_CANNOT_BE_PROCESSED

Host transmit:	**7F**	**80**	**01**	**09**	**35**	**82**
Slave Reply:	**7F**	**80**	**01**	**F5**	**3D**	**80**

<< back to index

Command	Code hex	Code decimal
Enable	0x0A	10

Implemented on	Encryption Required
NV200	**optional**

Description

This command will enable the SSP device for normal operation. For example, it will allow a banknote validator to commence validating banknotes entered into it's bezel.

For Image Capture equipment, the enable command allows faces to be detected and processed, as per the device's capabilities. For example, an Age Verification may be made of the person infront of the camera. The Enable command enables for a single measurement, and once complete (successfully or not) will then revert to disabled.

Packet examples

Single byte command with no parameters

Host transmit:	**7F**	**80**	**01**	**0A**	**3F**	**82**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

NV11 when note float is jammed/disconnected responds COMMAND_CANNOT_BE_PROCESSED

Host transmit:	**7F**	**80**	**01**	**0A**	**3F**	**82**
Slave Reply:	**7F**	**80**	**01**	**F5**	**3D**	**80**

<< back to index

Command	Code hex	Code decimal
Get Firmware Version	0x20	32

Implemented on	Encryption Required
NV200	**optional**

Description

Returns a variable length ASCII array containg the full firmware version of the attached device.

Packet examples

In this example, the firmware version of the device is: NV02004141498000

```
Host transmit:  7F  80  01  20  C0  02
 Slave Reply:   7F  80  11  F0  4E  56  30  32  30  30  34  31  34  31  34  39  38  30  30  30  DE  55
     ascii:                 .   N   V   0   2   0   0   4   1   4   1   4   9   8   0   0   0
```

<< back to index

Command	Code hex	Code decimal
Get Dataset Version	0x21	33

Implemented on	Encryption Required
NV200	**optional**

Description

Returns a varibale length ASCII array giving the installed dataset version of the device.

Packet examples

This example shows a device with dataset version EUR01610.

```
Host transmit:   7F  80  01  21  C5  82
 Slave Reply:    7F  80  09  F0  45  55  52  30  31  36  31  30  B8  2A
      ascii:                     .   E   U   R   0   1   6   1   0
```

<< back to index

Command	Code hex	Code decimal
Set Inhibits	0x02	2

Implemented on	Encryption Required
NV200	**optional**

Description

Sets the channel inhibit level for the device. Each byte sent represents 8 bits (channels of inhibit). The first byte is channels 1-8, second byte is 9-16 etc.

Nv200 has the option to send 1, 2 or 3 bytes to represent 8, 16 or 24 channels. The other BNV devices have the option of sending 1 or 2 bytes for 8 or 16 channel operation. Any channels not included in the request will be inhibited (eg. sending 1 byte inhibits channels 9+).

Set the bit low to inhibit all note acceptance on that channel, high to allow note acceptance.

Packet examples

Set channels 1-3 enabled, 4-16 inhibited

Host transmit:	**7F**	**80**	**03**	**02**	**07**	**00**	**2B**	**B6**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**		

16 channels enabled

Host transmit:	**7F**	**80**	**03**	**02**	**FF**	**FF**	**25**	**A4**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**		

<< back to index

Command	Code hex	Code decimal
Reject	0x08	8

Implemented on	Encryption Required
NV200	**optional**

Description

After a banknote validator device reports a valid note is held in escrow, this command may be sent to cause the banknote to be rejected back to the user.

Returns COMMAND_CANNOT_BE_PROCESSED if no note is in escrow.

Packet examples

Single byte command with no parameters

```
Host transmit:   7F  80  01  08  30  02
 Slave Reply:    7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Last Reject Code	0x17	23

Implemented on	Encryption Required
NV200	**optional**

Description

Returns a one byte code representing the reason the BNV rejected the last note. See Reject Code Table at the start of the manual for more information.

Packet examples

Note rejected due to a request by the host

```
Host transmit:   7F  80  01  17  71  82
 Slave Reply:    7F  80  02  F0  08  0C  20
```

<< back to index

Command	Code hex	Code decimal
Get Barcode Reader Configuration	0x23	35

Implemented on	Encryption Required
NV200	**optional**

Description

Returns the set-up data for the device bar code readers.

Responds (if supported) with five bytes of data formatted as:

byte	function	size
0	Generic OK	1
1	Bar code hardware status (0x00 = none, 0x01 = Top reader fitted, 0x02 = Bottom reader fitted, 0x03 = both fitted)	1
2	Readers enabled (0x00 = none, 0x01 = top, 0x02 = bottom, 0x03 = both)	1
3	Bar code format (0x01 = Interleaved 2 of 5)	1
4	Number of characters (Min 6 max 24)	1

Packet examples

Response for device with top and bottom readers fitted, both enabled, interleaved 2 of 5 with 18 chars

```
Host transmit:   7F  80  01  23  CA  02
 Slave Reply:    7F  80  05  F0  03  03  01  12  D5  58
```

<< back to index

Command	Code hex	Code decimal
Set Barcode Reader Configuration	0x24	36

Implemented on	Encryption Required
NV200	**optional**

Description

This command allows the host to set-up the bar code reader(s) configuration on the device.

Three bytes of data define the configuration:

byte	function	size
0	0x00 Enable none, 0x01 enable top, 0x02 = enable bottom, 0x03 = enable both	1
1	Bar code format (0x01 = Interleaved 2 of 5)	1
2	Number of characters (Min 6 Max 24)	1

Packet examples

Enable both readers with format interleaved 1 of 5 for 18 characters.

```
Host transmit:   7F  80  04  24  03  01  12  EC  D7
 Slave Reply:    7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Get Barcode Inhibit	0x25	37

Implemented on	Encryption Required
NV200	**optional**

Description

Command to return the current barcode/currency inhibit status.

If supported, responds with 1 byte bit register data:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
not used 1	not used 1	not used 1	not used 1	not used 1	not used 1	barcode read enable (0 = enabled)	currency read enable (0 = enabled)

FF (255) - Disable both currency and barcode

FE (254) - Disable Barcode and Enable Currency (Default)

FD (253) - Enable Barcode and Disable Currency

FC (252) - Enable both currency and barcode

Packet examples

A response from a device with bar code disabled, currency enabled

Host transmit:	**7F**	**80**	**01**	**25**	**DE**	**02**	
Slave Reply:	**7F**	**80**	**02**	**F0**	**FE**	**38**	**22**

<< back to index

Command	Code hex	Code decimal
Set Barcode Inhibit	0x26	38

Implemented on	Encryption Required
NV200	**optional**

Description

Sets up the bar code inhibit status register.

Send a single data bit register byte formatted as:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
not used 1	not used 1	not used 1	not used 1	not used 1	not used 1	barcode read enable (0 = enabled)	currency read enable (0 = enabled)

FF (255) - Disable both currency and barcode

FE (254) - Disable Barcode and Enable Currency (Default)

FD (253) - Enable Barcode and Disable Currency

FC (252) - Enable both currency and barcode

Packet examples

Shows a request to enabled bar code, disable currency on the device

```
Host transmit:   7F  80  02  26  FD  3E  D6
 Slave Reply:    7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Get Barcode Data	0x27	39

Implemented on	Encryption Required
NV200	**optional**

Description

Command to obtain last valid bar code ticket data, send in response to a bar code ticket validated event. This command will return a variable length data steam, a generic response (OK) followed by a status byte, a bar code data length byte, then a stream of bytes of the ticket data in ASCII.

Response is formatted as:

byte	function	size
0	Generic OK	1
1	Status (0=no valid data, 1=ticket in escrow, 2=ticket stacked, 3=ticket rejected)	1
2	data length (v)	1
3	variable length ASCII array of bar code data	v

Packet examples

shows ticket is in escrow with data length 6 and data 123456.

```
Host transmit:  7F  80  01  27  D1  82
  Slave Reply:  7F  80  09  F0  01  06  31  32  33  34  35  36  A1  05
        ascii:              .   .   .   1   2   3   4   5   6
```

<< back to index

Command	Code hex	Code decimal
Configure Bezel	0x54	84

Implemented on	Encryption Required
NV200	**optional**

Description

This command allows the host to configure a supported BNV bezel.

In NV200 firmware 4.28 an extra optional byte was added to specify the bezel type.

Command format:

byte	function	size
0	red pwm (0-255)	1
1	green pwm (0-255)	1
2	blue pwm (0-255)	1
3	Config 0 for volatile,1 - for non-volatile.	1
4	Optional Bezel Type (0 - Enable Solid Colour, 1 - Enable Flashing Colour, 2 - Disable Colour)	1

Packet examples

In this example, we want to enable solid red colour bezel fixed to EEPROM.

```
Host transmit:   7F  80  06  54  FF  00  00  01  00  FB  C9
Slave Reply:     7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Poll With Ack	0x56	86

Implemented on	Encryption Required
NV200	🔒 **yes**

Description

A command that behaves in the same way as the Poll command but with this command, some events will need to be acknowledged by the host using the EVENT ACK command (0x56). See the description of individual events to find out if they require acknowledgement.

If there is an event that requires acknowledgement the response will not change until the EVENT ACK command is sent and the BNV will not allow any further note actions until the event has been cleared by the EVENT ACK command. If this command is not supported by the slave device, then generic response 0xF2 will be returned and standard poll command (0x07) will have to be used.

Packet examples

Poll with ack sent and response is Stacking, Credit 01. This would require an ack afterwards otherwise the credit would repeat

Host transmit:	**7F**	**80**	**01**	**56**	**F7**	**83**			
Slave Reply:	**7F**	**80**	**04**	**F0**	**CC**	**EE**	**01**	**62**	**AA**

<< back to index

Command	Code hex	Code decimal
Event Ack	0x57	87

Implemented on	Encryption Required
NV200	🔒 **yes**

Description

This command will clear a repeating Poll ACK response and allow further note operations.

If no event currently requires acknowledgement a COMMAND_CANNOT_BE_PROCESSED response will be given.

Packet examples

Host transmit:	**7F**	**80**	**01**	**57**	**F2**	**03**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Get Counters	0x58	88

Implemented on	Encryption Required
NV200	**optional**

Description

A command to return a global note activity counter set for the slave device. The response is formatted as in the table below and the counter values are persistent in memory after a power down- power up cycle.

These counters are note set independent and will wrap to zero and begin again if their maximum value is reached. Each counter is made up of 4 bytes of data giving a max value of 4294967295.

Note Validator Response format:

byte	function	size
0	Generic OK	1
1	Number of counters in set	1
2	Stacked	4
6	Stored	4
10	Dispensed	4
14	Transferred to stack	4
18	Rejected	4

SH4 - SMART Coin System

Byte	Function	size
0	Generic OK	1
1	Number of counters in set	1
2	Coins paid out (includes to cashbox)	4
6	Coins paid in	4
10	Feeder Rejects	4
14	Hopper Jams	4
18	Feeder Jams	4
22	Fraud Attempts	4
26	Call Fails	4
30	Resets	4
34 (fw >= 1.26)	Coins sent to cashbox	4

SH3 - SMART Hopper

Byte	Function	size
0	Generic OK	1
1	Coins past sensors	4
5	Coins paid in	4
9	Coins paid out	4
13	Coins to cashbox	4
17	No of Payout requests	4
21	No of Float requests	4

Packet examples

Note Validator showing 5 counters: 17 stacked, 40 stored, 35 dispensed, 10 transferred and 16 rejects

```
Host transmit:  7F 80 01 58 D0 03
  Slave Reply:  7F 80 16 F0 05 11 00 00 00 28 00 00 00 23 00 00 00 0A 00 00 00 10 00 00
                00 72 B7
```

SH4 showing 8 counters: 110 hopper coins, 115 feeder coins, 6 feeeder rejects, 0 hopper jams, 0 feeder jams, 0 fraud attempts, 1 call fail and 3 resets

```
Host transmit:  7F 90 01 58 93 82
  Slave Reply:  7F 90 22 F0 08 6E 00 00 00 73 00 00 00 06 00 00 00 00 00 00 00 00 00 00
                00 00 00 00 00 01 00 00 00 03 00 00 00 0D B3
```

SH3 showing 2 coins past sensors, 3 coins paid in, 4 coins paid out, 5 coins to cashbox, 6 payout requests and 7 float requests

```
Host transmit:  7F 90 01 58 93 82
  Slave Reply:  7F 90 19 F0 02 00 00 00 03 00 00 00 04 00 00 00 05 00 00 00 06 00 00 00
                07 00 00 00 CB 58
```

<< back to index

Command	Code hex	Code decimal
Reset Counters	0x59	89

Implemented on	Encryption Required
NV200	**optional**

Description

Resets the note activity counters described in Get Counters command to all zero values.

Packet examples

Command format (no parameters) for acknowledged request.

Host transmit:	**7F**	**80**	**01**	**59**	**D5**	**83**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Set Generator	0x4A	74

Implemented on	Encryption Required
NV200	**optional**

Description

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Generator and must be a prime number. The slave will reply with OK or PARAMETER_OUT_OF_RANGE if the number is not prime.

Packet examples

In this example we are sending the prime number 982451653. This = 3A8F05C5 hex

```
Host transmit:   7F  80  09  4A  C5  05  8F  3A  00  00  00  00  B2  73
 Slave Reply:    7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Set Modulus	0x4B	75

Implemented on	Encryption Required
NV200	**optional**

Description

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Moduls and must be a prime number. The slave will reply with OK or PARAMETER_OUT_OF_RANGE if the number is not prime.

Packet examples

In this example we are sending the prime number 1287821. This = 13A68D hex

Host transmit:	**7F 80 09 4B 8D A6 13 00 00 00 00 00 6C F6**
Slave Reply:	**7F 80 01 F0 23 80**

<< back to index

Command	Code hex	Code decimal
Request Key Exchange	0x4C	76

Implemented on	Encryption Required
NV200	**optional**

Description

The eight data bytes are a 64 bit number representing the Host intermediate key. If the Generator and Modulus have been set the slave will calculate the reply with the generic response and eight data bytes representing the slave intermediate key. The host and slave will then calculate the key.

If Generator and Modulus are not set then the slave will reply FAIL.

Packet examples

An example of Host intermediate key of 7554354432121 = 6DEE29CC879 hex. Slave intermediate key = DB273CE5FA1B6823 hex

```
Host transmit:  7F  80  09  4C  79  C8  9C  E2  DE  06  00  00  9D  52
 Slave Reply:   7F  80  09  F0  23  68  1B  FA  E5  3C  27  DB  80  8A
```

<< back to index

Command	Code hex	Code decimal
Get Build Revision	0x4F	79

Implemented on	Encryption Required
NV200	**optional**

Description

A command to return the build revision information of a device.

For a single device the command returns 3 bytes of information representing the build of the product. For products made up of multiple devices (eg NV200 + Smart Payout) multiple revisions will be returned (3 bytes per product).

Byte 0 is the product type, next two bytes make up the revision number(0-65536).
For NV200 and Nv9usb the type byte is 0, for Note Float the byte is 7, and for SMART Payout the byte is 6.

Packet examples

This example is from an NV200 (issue 20) with payout attached (issue 21).

```
Host transmit:   7F  80  01  4F  A2  03
 Slave Reply:    7F  80  07  F0  00  14  00  06  15  00  0F  97
```

<< back to index

Command	Code hex	Code decimal
Set Baud Rate	0x4D	77

Implemented on	Encryption Required
NV200	**optional**

Description

This command has two data bytes to allow communication speed to be set on a device. Note that this command changes the **serial** baud rate.

byte	function	size
0	Required rate (0= 9600, 1=38400, 2= 115200)	1
1	Change persist (1=change will remain over reset, 0=rate sets to default after reset)	1

The device will respond with 0xF0 at the old baud rate before changing. Please allow a minimum of 100 millseconds before attempting to communicate at the new baud rate.

Packet examples

In this example, we want to temporarily set the speed to 38400 but to go back to the previous value when the unit is reset.

```
Host transmit:   7F  80  03  4D  01  00  E4  27
  Slave Reply:   7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Ssp Set Encryption Key	0x60	96

Implemented on	Encryption Required
NV200	🔒 **yes**

Description

A command to allow the host to change the fixed part of the eSSP key. The eight data bytes are a 64 bit number representing the fixed part of the key. This command must be encrypted.

byte	function	size
0	new fixed key 64 bit, 8 byte	8

Packet examples

Example to set new fixed key to 0x0123456701234567

```
Host transmit:   7F  80  09  60  67  45  23  01  67  45  23  01  BF  6F
 Slave Reply:    7F  80  01  F0  23  80
```

<< back to index

Command	Code hex	Code decimal
Ssp Encryption Reset To Default	0x61	97

Implemented on	Encryption Required
NV200	**optional**

Description

Resets the fixed encryption key to the device default. The device may have extra security requirements before it will accept this command (e.g. The Hopper must be empty) if these requirements are not met, the device will reply with Command Cannot be Processed. If successful, the device will reply OK, then reset. When it starts up the fixed key will be the default.

Packet examples

Command format (no parameters) for acknowledged request.

Host transmit:	**7F**	**80**	**01**	**61**	**46**	**03**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Ssp Download Data Packet	0x74	116

Implemented on	Encryption Required
NV200	**optional**

Description

Allows the download of a compatible SSP update file to a slave device. Please contact support@innovative-technology.com for more information.

Packet examples

<< back to index

Command	Code hex	Code decimal
Hold	0x18	24

Implemented on	Encryption Required
NV200	**optional**

Description

SSP banknote validators include a poll timeout of 10 seconds. If a new poll is not received within this time, then a note held in escrow will be rejected.
The host may require that the note is continued to be held, but a new poll would accept the note.
Sending this command (or any other command except poll) will reset the timeout and continue to hold the note in escrow until such time as either a reject or poll command is sent.

If there is no note in escrow then a COMMAND_CANNOT_BE_PROCESSED error will be sent.

Packet examples

Returns COMMAND CANNOTE BE PROCESSED if no note in escrow

Host transmit:	**7F**	**80**	**01**	**18**	**53**	**82**
Slave Reply:	**7F**	**80**	**01**	**F5**	**3D**	**80**

Holding a note that is in escrow

Host transmit:	**7F**	**80**	**01**	**18**	**53**	**82**
Slave Reply:	**7F**	**80**	**01**	**F0**	**23**	**80**

<< back to index

Command	Code hex	Code decimal
Setup Request	0x05	5

Implemented on	Encryption Required
NV200	**optional**

Description

Request the setup configuration of the device. Gives details about versions, channel assignments, country codes and values.

Each device type has a different return data format. Please refer to the device information table at the beginning of the manual for individual device data formats.

Packet examples

This example shows the data returned for a BNV with GBP dataset, firmware version 1.00, 3 channels GBP 5, GBP 10, GBP 20

```
Host transmit:  7F  80  01  05  1D  82
 Slave Reply:   7F  80  17  F0  00  30  31  30  30  47  42  50  00  00  01  03  05  0A  14  02  02  02  40  00
                00  05  61  81
    ascii:                  .   .   0   1   0   0   G   B   P   .   .   .   .   .   .   .   .   .   .   @   .
                .   .
```

This example shows the data returned for SMART Coin System with device type 9, firmware ver 121, GBP, protocol ver 7 and 8 denominations 1 - 200

```
Host transmit:  7F  90  02  05  05  28  5E
    ascii:          .   .   .   .   (   ^
 Slave Reply:   7F  90  30  F0  09  30  31  32  31  47  42  50  07  08  01  00  02  00  05  00  0A  00  14  00
                32  00  64  00  C8  00  47  42  50  47  42  50  47  42  50  47  42  50  47  42  50  47  42  50
                47  42  50  6C  DD
    ascii:                  .   .   0   1   2   1   G   B   P   .   .   .   .   .   .   .   .   .   .   .   .
                2   .   d   .   .   .   G   B   P   G   B   P   G   B   P   G   B   P   G   B   P   G   B   P
                G   B   P
```

<< back to index

Event	Code hex	Code decimal
Slave Reset	0xF1	241

Implemented on
NV200

Description

An event given when the device has been powered up or power cycled and has run through its reset process.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Poll returns slave reset event

```
Host transmit:   7F  80  01  07  12  02
Slave Reply:     7F  80  02  F0  F1  1A  22
```

<< back to index

Event	Code hex	Code decimal
Read	0xEF	239

Implemented on
NV200

Description

An event given when the BNV is reading a banknote.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**1**	**yes**	**no**

Additional infomation

If the event data byte is zero, then the note is in the process of being scanned and validated.

If the data byte value changes from zero to a vaule greater then zero, this indicates a valid banknote is now held in the escrow position. The byte value shows the channel of the banknote that has been validated. A poll command after this value has been given will cause the banknote to be accepted from the escrow position. The host can also issue a reject command at this point to reject the banknote back to the user. The Hold command may be used to keep the banknote in this position.

Protocol minimum version 9

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**7**	**yes**	**no**

Additional infomation

For the SMART Currency device only - 7 data bytes are given. If all bytes are zero then a banknote is in the process of being scanned and validated. Non zero show the country code and value of a validated banknote held in escrow.

data byte	function	size
0	3 byte ASCII code for country validated	3
3	4 byte code for banknote value	4

Packet examples

Poll response showing a biil being read but not yet validated.

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  EF  00  CF  CA
```

Poll response showing channel 3 bill held in escrow

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  EF  03  C5  CA
```

<< back to index

Event	Code hex	Code decimal
Note Credit	0xEE	238

Implemented on
NV200

Description

This event is generated when the banknote has been moved from the escrow position to a safe position within the validator system where the banknote cannot be retreived by the user.

At this point, it is safe for the host to use this event as it's 'Credit' point.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**1**	**yes**	**yes**

Additional infomation

The data byte indicates the dataset channel of the banknote to be credited.

Protocol minimum version 9

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**7**	**yes**	**no**

Additional infomation

For the SMART Currency device only - 7 data bytes are given showing the country code and value of a Credited banknote.

data byte	function	size
0	3 byte ASCII code for country validated	3
3	4 byte code for banknote value	4

Packet examples

Poll response showing bill credit channel 4

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  EE  04  D7  CC
```

<< back to index

Event	Code hex	Code decimal
Rejecting	0xED	237

Implemented on
NV200

Description

A bill is in the process of being rejected back to the user by the Banknte Validator.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**yes**	**no**

Packet examples

Poll response showing bill rejecting

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  ED  51  A2
```

<< back to index

Event	Code hex	Code decimal
Rejected	0xEC	236

Implemented on
NV200

Description

A bill has been rejected back to the user by the Banknote Validator.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Poll response showing bill rejected by the validator.

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  EC  54  22
```

<< back to index

Event	Code hex	Code decimal
Stacking	0xCC	204

Implemented on
NV200

Description

The bill is currently being moved from escrow into the device. The Stacked or Stored event will be given when this operation completes depending on where the note ended up.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**yes**	**no**

Packet examples

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  CC  97  A2
```

<< back to index

Event	Code hex	Code decimal
Stacked	0xEB	235

Implemented on
NV200

Description

A bill has been transported trough the banknote validator and is in it's stacked position.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Poll response showing stacked bill seen

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  EB  45  A2
```

<< back to index

Event	Code hex	Code decimal
Unsafe Jam	0xE9	233

Implemented on
NV200

Description

A bill has been detected as jammed during it's transport through the validator. An unsafe jam indicates that this bill may be in a position when the user could retrieve it from the validator bezel.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Error	**0**	**yes**	**no**

Packet examples

Poll response showing unsafe bill jam detected

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  E9  4A  22
```

<< back to index

Event	Code hex	Code decimal
Disabled	0xE8	232

Implemented on
NV200

Description

A disabled event is given in response to a poll command when a device has been disabled by the host or by some other internal function of the device.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Response to poll showing disabled event

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  E8  4F  A2
```

<< back to index

Event	Code hex	Code decimal
Fraud Attempt	0xE6	230

Implemented on
NV200

Description

The validator system has detected an attempt to manipulate the coin/banknote in order to fool the system and register credits with no money added.

Please note the event data reported is different if the unit is SMART Hopper 3 or SMART Hopper 4 / SMART System (see event data below).

To get the specific calibration error in SMART Hopper 4 / SMART System an expansion command is available, please contact ITL support for further information.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Fraud	**1**	**no**	**yes**

Additional infomation

The data byte indicates the dataset channel of the banknote that is being tampeted with. A zero indicates that the channle is unknown.

Protocol minimum version 5

Type	Data size (bytes)	Repeat	Poll with Ack
Fraud	**4**	**yes**	**yes**

Additional infomation

Event data for SMART Hopper 4 / SMART System when the protocol version is below 6. The 4 bytes represent the value dispensed/floated up to the fraud condition.

Protocol minimum version 6

Type	Data size (bytes)	Repeat	Poll with Ack
Fraud	**variable**	**yes**	**yes**

Additional infomation

Event data for SMART Hopper 4 / SMART System when the protocol version is the same or above 6. An array of data giving the dispensed/floated value at the fraud point for each of the countries supported in the dataset. The first byte gives the number of countries in the set then a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed/floated up to this point	4
5	country	3
...	repeat above block for each country in set	...

Packet examples

Poll response showing fraud attempt seen on channel 2

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  E6  02  C0  7C
```

For SMART Hopper 4 / SMART System with protocol version 6 poll response showing 15.30 EUR to the fraud attempt point.

```
Host transmit:   7F  90  01  07  51  83
 Slave Reply:    7F  90  0A  F0  E6  01  FA  05  00  00  45  55  52  B6  64
```

<< back to index

Event	Code hex	Code decimal
Stacker Full	0xE7	231

Implemented on
NV200

Description

Event in response to poll given when the device has detected that the stacker unit has stacked it's full limit of banknotes.

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Poll response showing stacker full

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  E7  6D  A2
```

<< back to index

Event	Code hex	Code decimal
Note Cleared From Front	0xE1	225

Implemented on
NV200

Description

During the device power-up sequence a bill was detected as being in the note path. This bill is then rejected from the device via the bezel and this event is issued. If the bill value is known then the channel number is given in the data byte, otherwise the data byte will be zero value.

Packet examples

Poll response showing unknown bill rejected from the front at power-up

Host transmit:	**7F**	**80**	**01**	**07**	**12**	**02**		
Slave Reply:	**7F**	**80**	**03**	**F0**	**E1**	**00**	**CC**	**6E**

<< back to index

Event	Code hex	Code decimal
Note Cleared Into Cashbox	0xE2	226

Implemented on
NV200

Description

During the device power-up sequence a bill was detected as being in the stack path. This bill is then moved into the device cashbox and this event is issued. If the bill value is known then the channel number is given in the data byte, otherwise the data byte will be zero value.

Protocol minimum version 5

Type	Data size (bytes)	Repeat	Poll with Ack
Pay-in	**1**	**no**	**yes**

Packet examples

Poll response showing a channel 2 bill moved to the cashbox at power-up

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  03  F0  E2  02  C3  E4
```

<< back to index

Event	Code hex	Code decimal
Cashbox Removed	0xE3	227

Implemented on
NV200

Description

The system has detected that the cashbox unit has been removed from it's working position.

The system will remain disabled for bill entry until the cashbox unit is replaced into it's working position.

Protocol minimum version 5

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**yes**	**no**

Packet examples

Poll response showing cashbox removed

```
Host transmit:   7F  80  01  07  12  02
Slave Reply:     7F  80  02  F0  E3  76  22
```

<< back to index

Event	Code hex	Code decimal
Cashbox Replaced	0xE4	228

Implemented on
NV200

Description

The device cashbox box unit has been detected as replaced into it's working position.

The validator will re-enable if it has not already been disabled by the host system.

Protocol minimum version 5

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

Poll response showing cashbox replaced

```
Host transmit:   7F  80  01  07  12  02
Slave Reply:     7F  80  02  F0  E4  67  A2
```

<< back to index

Event	Code hex	Code decimal
Barcode Ticket Validated	0xE5	229

Implemented on
NV200

Description

A barcode ticket has been scanned and identified by the system and is currently held in the escrow position.

The host can send the Get Barcode Data command to retrive the number of the ticket scanned. The host can then send a Reject or Poll command to reject or accept the ticket as required.

Packet examples

Poll response showing bar code held in escrow

Host transmit:	**7F**	**80**	**01**	**07**	**12**	**02**	
Slave Reply:	**7F**	**80**	**02**	**F0**	**E5**	**62**	**22**

<< back to index

Event	Code hex	Code decimal
Barcode Ticket Ack	0xD1	209

Implemented on
NV200

Description

The device has moved the barcode ticket into the cashbox (equivalent to Note Credit event for a bank note)

Protocol minimum version 4

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**yes**

Packet examples

Poll response showing bar code ticket ack

Host transmit:	**7F**	**80**	**01**	**07**	**12**	**02**	
Slave Reply:	**7F**	**80**	**02**	**F0**	**D1**	**D9**	**A2**

<< back to index

Event	Code hex	Code decimal
Note Path Open	0xE0	224

Implemented on
NV200

Description

The device has detected that it's note path has been opened. The device will be disabled for bill entry until the note path is re-closed.

Protocol minimum version 6

Type	Data size (bytes)	Repeat	Poll with Ack
Error	**0**	**yes**	**no**

Packet examples

Poll response showing note path open

```
Host transmit:   7F  80  01  07  12  02
Slave Reply:     7F  80  02  F0  E0  7C  22
```

<< back to index

Event	Code hex	Code decimal
Channel Disable	0xB5	181

Implemented on
NV200

Description

The device has had all its note channels inhibited and has become disabled for note insertion. Use the Set Inhibits command to enable some notes to remove this event.

Protocol minimum version 7

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**no**	**no**

Packet examples

```
Host transmit:   7F  80  01  07  12  02
 Slave Reply:    7F  80  02  F0  B5  82  23
```

<< back to index

Event	Code hex	Code decimal
Initialising	0xB6	182

Implemented on
NV200

Description

This event is given only when using the Poll with ACK command (though it doesn't need an event ACK to be cleared as other Poll with Ack commands). It is given when the BNV is powered up and setting its sensors and mechanisms to be ready for Note acceptance. When the event response does not contain this event, the BNV is ready to be enabled and used.

Protocol minimum version 7

Type	Data size (bytes)	Repeat	Poll with Ack
Status	**0**	**yes**	**no**

Additional infomation

This event is only given when using the Poll With Ack command (but doesn't need an Event Ack sent afterwards).

Packet examples

Host transmit:	**7F**	**80**	**01**	**07**	**12**	**02**	
Slave Reply:	**7F**	**80**	**02**	**F0**	**B6**	**88**	**23**