Protocol Manual

CC2

SMART HOPPER, SMART PAYOUT, SMART SYSTEM

version GA863_1_2_50A

A.u.S. Spielgeräte GesmbH
Scheydgasse 48   A-1210 Wien
Tel. +43-1-271 66 00 Fax.+43-1-271 66 00 75
E-mail: verkauf@aus.at
http://www.aus.at
Öffnungszeiten: Mo-Fr. 9-18 Uhr

Contents

Descriptions

Command/Event Tables

Commands

Events

<< back to index

Introduction

This manual describes the operation of the ITL Protocol **CC2**.

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.

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

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 optionaly ensured through the use of 16 bit CRC checksums on packets.

Each CC2 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.

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

Values are represented as 32 bit unsigned integer (4 bytes) and in the lowest value of currency. For example 125.65 would be 0x00003115.
When sending or receiving a value the Least significant byte is sent first. So in this example [0x15] [0x31] [0x00] [0x00] will be sent. When using commands with multicurrency support all values will be followed by a 3 byte ASCII Currency code (e.g. 069 085 082 for EUR)

<< back to index

Pay-out Encryption

The pay-out devices support various levels of encryption. These options are configured on an individual device off-line by PC tools.

Options:

- Level 0 No encryption All packets are plain with 8-bit checksum. No payout commands are encrypted. THIS IS THE LOWEST LEVEL OF SECURITY AND THE USER WOULD NEED TO CONSIDER THE POSSIBILITY OF BUS HI-JACK FRAUDS WITH THIS OPTION.
- Level 1 Type 1 security bytes for payout commands.

- Level 2 Type 2 security bytes for payout commands.
- DES Encryption for payout commands.
- Packet Encryption Options:
- BNV packet encryption.
- All packets are encrypted using Money Controls BNV security algorithm
- Packet checksum
- Option of using 8bit addition or 16bit CRC checksum.

<< back to index

SMART Hopper

SMART Hopper is a coin payout device capable of discriminating and paying out multi-denominations of stored coins from its internal storage hopper.
Coins added to the hopper can be designated to be routed to an external cashbox on detection or recycled and stored in the hopper unit to be available for a requested payout.
SMART Hopper also supports the addition of a connected cctalk™ or eSSP™ coin mechanism which will automatically add its validated coins to the SMART Hopper system levels.

Note that payout values are in terms of the of the penny value of that currency. So for 5.00, the value sent and returned by the hopper would be 500. All transactions with a SMART hopper should be encrypted to prevent dispense commands being recorded and replayed by an external device.

<< back to index

SMART Payout

The Smart Payout is an extension of a banknote validator, all commands are sent to the validator using its address (0x00). Information on the types of note that can be handled is obtained from the standard note validator commands.
Note that payout values are in terms of the penny value of that currency. So for 5.00, the value sent and returned by the payout would be 500.

The host simply has to tell the unit the value it wishes to dispense. The unit will manage which notes are stored to be used for payout and their location to minimise the payout time, and which notes, of the type enable for storage, are sent to the stacker. This is the recommended mode of operation.

<< back to index

SMART System

The Smart System device is a multi-coin pay-in, pay-out system with detachable fast coin pay-in feeder.

Coins fed into the pay-in head will be validated and counted and recognised coins are routed to the attached hopper while rejected coins are fed out of the front of the system.

Coin hopper levels are adjusted internally.

The system can function as a stand-alone hopper payout system if the pay-in feeder head is removed.

<< back to index

SMART HOPPER Command Table

	Header code (hex)	dec
Reset Device	0x01	1
Payout Amount	0x16	22
Payout Amount (cur)	0x27	39
Simple Poll	0xFE	254
Request Comms Revision	0x04	4
Set Routing	0x14	20
Address Poll	0xFD	253
Request Manufacturer Id	0xF6	246
Request Equipment Category Id	0xF5	245
Request Product Code	0xF4	244
Request Serial Number	0xF2	242
Request Software Revision	0xF1	241
Set Master Inhibit Status	0xE4	228
Get Master Inhibit Status	0xE3	227
Request Data Storage Capability	0xD8	216
Request Last Mod Date	0xC3	195
Request Build Code	0xC0	192
Request Address Mode	0xA9	169
Pump Rng	0xA1	161
Request Cipher Key	0xA0	160
Switch Encryption Code	0x89	137
Store Encryption Code	0x88	136
Request Encryption Support	0x6F	111
Switch Des Key	0x6E	110
Request Encrypted Status	0x6D	109
Get Inhibit Peripheral Device Value	0x35	53
Get Cashbox Operation Data	0x34	52
Smart Empty	0x33	51
Set Inhibit Peripheral Device Value	0x32	50
Get Peripheral Device Master Inhibit	0x31	49
Set Peripheral Device Master Inhibit	0x30	48
Request Status (cur)	0x2F	47
Get Device Setup (cur)	0x2E	46
Float By Denomination (cur)	0x2D	45
Payout By Denomination (cur)	0x2C	44
Set Denomination Amount (cur)	0x2B	43
Get Denomination Amount (cur)	0x2A	42
Get Minimum Payout (cur)	0x29	41
Float Amount (cur)	0x28	40
Get Routing (cur)	0x26	38
Set Routing (cur)	0x25	37
Run Unit Calibration	0x22	34
Float By Denomination	0x21	33
Payout By Denomination	0x20	32
Get Payout Options	0x1F	31
Set Payout Options	0x1E	30
Request Status	0x1D	29
Get Device Setup	0x1C	28
Set Denomination Amount	0x1B	27
Get Denomination Amount	0x1A	26
Get Minimum Payout	0x19	25
Empty	0x18	24
Float Amount	0x17	23
Get Routing	0x15	21
Halt	0x40	64

SMART HOPPER Event Table

	Header code (hex)	dec
Dispensing	0x01	1
Idle	0x00	0
Dispensed	0x02	2
Coins Low	0x03	3
Empty	0x04	4
Halted	0x06	6
Floating	0x07	7
Floated	0x08	8
Timeout	0x09	9
Cashbox Paid	0x0C	12
Coin Credit	0x0D	13
Emptying	0x0E	14
Emptied	0x0F	15
Fraud Attempt	0x10	16
Disabled	0x11	17
Slave Reset	0x13	19
Lid Open	0x21	33
Lid Closed	0x22	34
Calibration Fault	0x24	36
Attached Mech Jam	0x25	37
Attached Mech Open	0x26	38
Smart Emptying	0x27	39
Smart Emptied	0x28	40
Incomplete Payout	0x0A	10
Incomplete Float	0x0B	11

<< back to index

SMART PAYOUT Command Table

	Header code (hex)	dec
Reset Device	0x01	1
Payout Amount	0x16	22
Payout Amount (cur)	0x27	39
Simple Poll	0xFE	254
Request Comms Revision	0x04	4
Set Routing	0x14	20
Address Poll	0xFD	253
Address Clash	0xFC	252
Address Change	0xFB	251
Address Random	0xFA	250
Request Polling Priority	0xF9	249
Request Manufacturer Id	0xF6	246
Request Equipment Category Id	0xF5	245
Request Product Code	0xF4	244
Request Serial Number	0xF2	242
Request Software Revision	0xF1	241
Set Note Inhibit Channels	0xE7	231
Request Note Channel Inhibits	0xE6	230
Set Master Inhibit Status	0xE4	228
Get Master Inhibit Status	0xE3	227
Request Data Storage Capability	0xD8	216
Request Option Flags	0xD5	213
Request Last Mod Date	0xC3	195
Request Build Code	0xC0	192
Request Address Mode	0xA9	169
Pump Rng	0xA1	161
Request Cipher Key	0xA0	160
Read Buffered Bill Events	0x9F	159
Request Bill Id	0x9D	157
Request Country Scaling Factor	0x9C	156
Request Bill Position	0x9B	155
Route Bill	0x9A	154
Modify Bill Operating Mode	0x99	153
Request Bill Operating Mode	0x98	152
Request Currency Revision	0x91	145
Switch Encryption Code	0x89	137
Store Encryption Code	0x88	136
Read Barcode Data	0x81	129
Request Encryption Support	0x6F	111
Switch Des Key	0x6E	110
Request Encrypted Status	0x6D	109
Request Status (cur)	0x2F	47
Get Device Setup (cur)	0x2E	46
Float By Denomination (cur)	0x2D	45
Payout By Denomination (cur)	0x2C	44
Get Denomination Amount (cur)	0x2A	42
Get Minimum Payout (cur)	0x29	41
Float Amount (cur)	0x28	40
Get Routing (cur)	0x26	38
Set Routing (cur)	0x25	37
Set Bezel Mode	0x23	35
Run Unit Calibration	0x22	34
Float By Denomination	0x21	33
Payout By Denomination	0x20	32
Get Payout Options	0x1F	31
Set Payout Options	0x1E	30
Request Status	0x1D	29
Get Device Setup	0x1C	28
Get Denomination Amount	0x1A	26
Get Minimum Payout	0x19	25
Empty	0x18	24
Float Amount	0x17	23
Get Routing	0x15	21
Halt	0x40	64

SMART PAYOUT Event Table

	Header code (hex)	dec
Dispensing	0x01	1
Note Read	0x14	20
Idle	0x00	0
Dispensed	0x02	2
Halted	0x06	6
Floating	0x07	7
Floated	0x08	8
Timeout	0x09	9
Emptying	0x0E	14
Emptied	0x0F	15
Fraud Attempt	0x10	16
Disabled	0x11	17
Note Stored	0x12	18
Slave Reset	0x13	19
Note Credit	0x15	21
Note Rejecting	0x16	22
Rejected	0x17	23
Stacking	0x18	24
Stacked	0x19	25
Note Path Jam	0x1A	26
Note Stack Jam	0x1B	27
Bill From Front At Start	0x1C	28
Bill Stacked At Start	0x1D	29
Cashbox Full	0x1E	30
Cashbox Removed	0x1F	31
Cashbox Replaced	0x20	32
Smart Emptying	0x27	39
Smart Emptied	0x28	40
Barcode Escrow	0x34	52
Barcode Stacked	0x35	53
Bill Held In Bezel	0x39	57
Incomplete Payout	0x0A	10
Incomplete Float	0x0B	11
Bill Stored At Startup	0x3C	60
Error During Payout	0x30	48
Payout Jam Recovery	0x31	49
Startup Initialisation Active	0x32	50
All Channels Inhibited	0x33	51

<< back to index

SMART SYSTEM Command Table

	Header code (hex)	dec
Reset Device	0x01	1
Payout Amount	0x16	22
Payout Amount (cur)	0x27	39
Simple Poll	0xFE	254
Request Comms Revision	0x04	4
Set Routing	0x14	20
Address Poll	0xFD	253
Request Manufacturer Id	0xF6	246
Request Equipment Category Id	0xF5	245
Request Product Code	0xF4	244
Request Serial Number	0xF2	242
Request Software Revision	0xF1	241
Set Master Inhibit Status	0xE4	228
Get Master Inhibit Status	0xE3	227
Request Data Storage Capability	0xD8	216
Request Last Mod Date	0xC3	195
Request Build Code	0xC0	192
Request Address Mode	0xA9	169
Pump Rng	0xA1	161
Request Cipher Key	0xA0	160
Switch Encryption Code	0x89	137
Store Encryption Code	0x88	136
Request Encryption Support	0x6F	111
Switch Des Key	0x6E	110
Request Encrypted Status	0x6D	109
Get Inhibit Peripheral Device Value	0x35	53
Get Cashbox Operation Data	0x34	52
Smart Empty	0x33	51
Set Inhibit Peripheral Device Value	0x32	50
Get Peripheral Device Master Inhibit	0x31	49
Set Peripheral Device Master Inhibit	0x30	48
Request Status (cur)	0x2F	47
Get Device Setup (cur)	0x2E	46
Float By Denomination (cur)	0x2D	45
Payout By Denomination (cur)	0x2C	44
Set Denomination Amount (cur)	0x2B	43
Get Denomination Amount (cur)	0x2A	42
Get Minimum Payout (cur)	0x29	41
Float Amount (cur)	0x28	40
Get Routing (cur)	0x26	38
Set Routing (cur)	0x25	37
Run Unit Calibration	0x22	34
Float By Denomination	0x21	33
Payout By Denomination	0x20	32
Get Payout Options	0x1F	31
Set Payout Options	0x1E	30
Request Status	0x1D	29
Get Device Setup	0x1C	28
Set Denomination Amount	0x1B	27
Get Denomination Amount	0x1A	26
Get Minimum Payout	0x19	25
Empty	0x18	24
Float Amount	0x17	23
Get Routing	0x15	21
Request Hopper Status	0xA6	166
Payout Amount By Denomination	0x24	36
Set Cashbox Payout Limits	0x36	54
Halt	0x40	64

SMART SYSTEM Event Table

	Header code (hex)	dec
Dispensing	0x01	1
Idle	0x00	0
Dispensed	0x02	2
Coins Low	0x03	3
Empty	0x04	4
Halted	0x06	6
Floating	0x07	7
Floated	0x08	8
Timeout	0x09	9
Cashbox Paid	0x0C	12
Coin Credit	0x0D	13
Emptying	0x0E	14
Emptied	0x0F	15
Fraud Attempt	0x10	16
Disabled	0x11	17
Slave Reset	0x13	19
Calibration Fault	0x24	36
Attached Mech Jam	0x25	37
Attached Mech Open	0x26	38
Smart Emptying	0x27	39
Smart Emptied	0x28	40
Multiple Value Added	0x36	54
Peripheral Error	0x37	55
Peripheral Device Disabled	0x38	56
Value Pay-in	0x3A	58
Incomplete Payout	0x0A	10
Incomplete Float	0x0B	11
Device Full	0x3B	59

<< back to index

Command	Code hex	Code decimal
Reset Device	0x01	1

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command causes the device to carry out a full reset. The Device sends a positive acknowledgement immediately before making the reset.

Packet examples

Reset command

Host transmit:	**28 00 01 01 D6**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Payout Amount	0x16	22

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be paid from the device. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command. Non-security levels just return an ACK.

Command data format(no security bytes shown)

byte	function	size
0	value to pay	4

Returns NAK with error code for request failures:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Payout Amount (cur)	0x27	39

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be paid from the device. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command. Non-security levels just return an ACK.

Command data format(no security bytes shown)

byte	function	size
0	value to pay	4
4	country code	3

Returns NAK with error code for request failures:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Simple Poll	0xFE	254

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Command to check the correct operation of communication and to confirm the presence in the bus of a device.

If no reply is received to the request sent (reception timeout in Machine), it will indicate that the device is faulty or not connected. All the cctalk peripherals must respond to a Simple Poll, regardless of the cctalk communication protocol level that has been implemented.

Packet examples

Simple poll command

Host transmit:	**28 00 01 FE D9**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Request Comms Revision	0x04	4

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

As a reply to this command, the Smart Payout or Hopper sends the implementation level of the cctalk protocol and the communication software version. E.g. Data 1 = 0 Data 2 = 1 Data 3 = 2 Data 4 = 0 Gives a revision of 1.2.0.

This version ties in with the version of the specification document and allows updates and changes to be tracked.

Packet examples

Example show reivsion 1.2.6

```
Host transmit:  28 00 01 04 D3
  Slave Reply:  01 03 28 00 01 02 06 CB
```

<< back to index

Command	Code hex	Code decimal
Set Routing	0x14	20

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A command to control the route of a denomination entered in to the device. Money can either be stored in the device available for payout or sent to an external cashbox.

For routes to cashbox - In the case of the ***Smart Payout,*** notes entered will be routed straight to the cashbox; with the ***Smart Hopper/Smart System*** coins will be routed to the cashbox as they are detected by the hopper discrimination system.

Table show command data format.

byte	function	size
0	requested route (0 = payout, 1= cashbox)	1
1	value	4

The device will reply with an ACK for successful operation or NAK for command failure.

Packet examples

Example showing 5.00 bill routed to the cashbox

```
Host transmit:  28 05 01 14 01 F4 01 00 00 C8
  Slave Reply:  01 00 28 00 D7
```

Example showing 5.00 bill routed to the cashbox with NAK response

```
Host transmit:  28 05 01 14 01 F4 01 00 00 C8
  Slave Reply:  01 00 28 05 D2
```

<< back to index

Command	Code hex	Code decimal
Address Poll	0xFD	253

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The host sends this command as a broadcast address packet (0 destination address). The device responses with a single byte containing its address with a series of delays:

Disable rx port

Delay (4 * addr) ms

Send [addr]

Delay 1200 - (4 * addr) ms

Enable rx port

Packet examples

<< back to index

Command	Code hex	Code decimal
Address Clash	0xFC	252

Implemented on
SMART PAYOUT

Description

This command is transmitted to a specified address. It attempts to determine if one or more devices share the same address. The device returns a single byte of address data after a random delay:

Slave Response Algorithm r = rand(256) // random value in the range 0 to 255

Disable rx port Delay (4 * r) ms Send [addr]

Delay 1200 - (4 * r) ms

Enable rx port

Packet examples

Host transmit:	**28 00 01 FC DB**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Address Change	0xFB	251

Implemented on
SMART PAYOUT

Description

This command allows the addressed device to have its address changed for subsequent commands. The host sends 1 data byte, the value of which is the new address.

It is a good idea to make sure that 2 devices do not share the same address before sending this command.

A full ACK message is returned. Note the ACK is sent back from the original address, not the changed address. In other words, the change to the ccTalk address field is done after the ACK is returned rather than before.

Packet examples

Change address to 4

```
Host transmit:  28 01 01 FB 04 D7
  Slave Reply:  01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Address Random	0xFA	250

Implemented on
SMART PAYOUT

Description

This command allows the addressed device to have its address changed to a random value. This is the escape
route when you find out that one or more devices share the same address. A full ACK message is returned.

Packet examples

```
Host transmit:  28  00  01  FA  DD
  Slave Reply:  01  00  28  00  D7
```

<< back to index

Command	Code hex	Code decimal
Request Polling Priority	0xF9	249

Implemented on
SMART PAYOUT

Description

This is an indication by a device of the recommended polling interval for buffered credit information. Polling a device at an interval longer than this may result in lost credits.

[units] 0 - special case, see below 1 - ms 2 - x10 ms 3 - seconds 4 - minutes 5 - hours 6 - days 7 - weeks 8 - months 9 - years

Packet examples

Example showing polling priority of 200ms

```
Host transmit:   28  00  01  F9  DE
  Slave Reply:   01  02  28  00  02  14  BF
```

<< back to index

Command	Code hex	Code decimal
Request Manufacturer Id	0xF6	246

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the ASCII code for the manufacturer of the device. In this case 'ITL'

Packet examples

Returns ITL

```
Host transmit:  28  00  01  F6  E1
  Slave Reply:  01  03  28  00  49  54  4C  EB
      ascii:                     I   T   L
```

<< back to index

Command	Code hex	Code decimal
Request Equipment Category Id	0xF5	245

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Returns the type of connected device as ascii array.

Packet examples

Example showing data response for SMART_HOPPER.

```
Host transmit:  03  00  01  F5  07
  Slave Reply:  01  0C  03  00  53  4D  41  52  54  5F  48  4F  50  50  45  52  3C
        ascii:                   S   M   A   R   T   _   H   O   P   P   E   R
```

<< back to index

Command	Code hex	Code decimal
Request Product Code	0xF4	244

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the device product code. The complete identification of the product can be determined by the use of the [Request product code] command followed by the [Request build code] command.

Packet examples

Example response showing SMART Hopper code SH3

```
Host transmit:  03 00 01 F4 08
  Slave Reply:  01 03 03 00 53 48 33 2B
```

<< back to index

Command	Code hex	Code decimal
Request Serial Number	0xF2	242

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

In reply to this command, the Device sends the serial number of the device in a 3-byte big endian array.

Packet examples

Example showing serial number return from a device with code 4321432

```
Host transmit:  28  00  01  F2  E5
  Slave Reply:  01  03  28  00  41  F0  98  0B
```

<< back to index

Command	Code hex	Code decimal
Request Software Revision	0xF1	241

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the full software revision of the device as an ASCII string.

Packet examples

Example showing software revision of NV02004232741000

```
Host transmit:  28  00  01  F1  E6
  Slave Reply:  01  10  28  00  4E  56  30  32  30  30  34  32  33  32  37  34  31  30  30  30  6A
       ascii:                   N   V   0   2   0   0   4   2   3   2   7   4   1   0   0   0
```

<< back to index

Command	Code hex	Code decimal
Set Note Inhibit Channels	0xE7	231

Implemented on
SMART PAYOUT

Description

This command sets the inhibit status of each of the 16 available channels on a bill acceptor device.

Command has two byte bit field. Each bit represents a bill channel. bit 0 = channel1 to bit 15 = channel 16. Set to 0 to inhibit channel, 1 to enable channel.

Packet examples

Example showing a command set to enable channels 1,2 and 3 only.

```
Host transmit:   28 02 01 E7 07 00 E7
  Slave Reply:   01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Request Note Channel Inhibits	0xE6	230

Implemented on
SMART PAYOUT

Description

This command returns the status of the two byte inhibit register for the bill channels of a note accepting device.

Response returns the two byte inihibit register. bit 0 = channel 1 to bit 15 = channel 16.

Packet examples

Example showing a register setup of channel 2,3 and 4 enabled and all others inhibited.

Host transmit:	**28 00 01 E6 F1**
Slave Reply:	**01 02 28 00 0E 00 C7**

<< back to index

Command	Code hex	Code decimal
Set Master Inhibit Status	0xE4	228

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A command to globally enable or disable the payout device or bill validator for payout/paying in operations.
This value is stored in volatile ram and will be set to disabled state after a reset.

Command has 1 data byte which is a bit field. Bit 0 controls the Master inhibit state (0 = disable, 1 = enable) Bits 1-7 are not used.

Packet examples

Example showing a command to set the master inhibit state to enable.

```
Host transmit:  28  01  01  E4  01  F1
  Slave Reply:  01  00  28  00  D7
```

<< back to index

Command	Code hex	Code decimal
Get Master Inhibit Status	0xE3	227

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the current status of the Master inhibit value from the device.

Response contains the master inhibit status register byte. Bit 0 gives the status: 0 is disabled, 1 is enabled. Bits 1 to 7 are not used.

Packet examples

This response shows the master inhibit set disabled.

Host transmit:	**28 00 01 E3 F4**
Slave Reply:	**01 01 28 00 00 D6**

<< back to index

Command	Code hex	Code decimal
Request Data Storage Capability	0xD8	216

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A command to return the data storage capability of the device.

This command is included for system compatibility. **ITL device products return all 0 for this.**

Packet examples

This command always returns 5 zero bytes for ITL products

```
Host transmit:  28 00 01 D8 FF
  Slave Reply:  01 05 28 00 00 00 00 00 00 D2
```

<< back to index

Command	Code hex	Code decimal
Request Option Flags	0xD5	213

Implemented on
SMART PAYOUT

Description

This command returns a one byte bit field register formatted as: 0 stacker, 1 escrow, 2 individual bill accept counters, 3 individual error counters, 4 non-volatile counters, 5 bill teach facility, 6 bill security tuning, 7 remote bill programming. If the bit is set (1) the option is supported.

Packet examples

This example shows a response of a device supporting stacker and escrow functionality.

Host transmit:	**28 00 01 D5 02**
Slave Reply:	**01 01 28 00 03 D3**

<< back to index

Command	Code hex	Code decimal
Request Last Mod Date	0xC3	195

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Returns an encoded date array showing the last modification of this device.

Base year for ITL SMART products is 2009

Packet examples

Example showing date 1st Jan 1 year after base year.

```
Host transmit:   28  00  01  C3  14
  Slave Reply:   01  02  28  00  11  01  C3
```

<< back to index

Command	Code hex	Code decimal
Request Build Code	0xC0	192

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return an ASCII array describing the build version of the device.

Packet examples

Example response showing build code: standard

```
Host transmit:  28  00  01  C0  17
  Slave Reply:  01  08  28  00  73  74  61  6E  64  61  72  64  7E
      ascii:                     s   t   a   n   d   a   r   d
```

<< back to index

Command	Code hex	Code decimal
Request Address Mode	0xA9	169

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the mode in which the cctalk address is stored and if it can be changed by serial command.

Returns a bit register configured as:

B0 - Address is stored in ROM

B1 - Address is stored in RAM

B2 - Address is stored in EEPROM or NV memory

B3 - Address selection via interface connector

B4 - Address selection via PCB links

B5 - Address selection via switch

B6 - Address may be changed with serial commands (volatile)

B7 - Address may be changed with serial commands (non-volatile)

Packet examples

Example response showing address is stored in EEPROM but not changeable by serial command.

Host transmit:	**28 00 01 A9 2E**
Slave Reply:	**01 01 28 00 04 D2**

<< back to index

Command	Code hex	Code decimal
Pump Rng	0xA1	161

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command is part of the security level payouts. No further details are given here.

Packet examples

<< back to index

Command	Code hex	Code decimal
Request Cipher Key	0xA0	160

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command is used in security level payouts. No further details are given here.

Packet examples

<< back to index

Command	Code hex	Code decimal
Read Buffered Bill Events	0x9F	159

Implemented on
SMART PAYOUT

Description

This command gives of the last 5 bill events on a bill accepting device.

For CC2 protocol SMART Payout, this command always returns zeros. This is done for compatibility with some host systems. The host should use the Get Status (029) command for CC2 events.

Packet examples

Returns zero for CC2 device

```
Host transmit:  28 00 01 9F 38
  Slave Reply:  01 0B 28 00 00 00 00 00 00 00 00 00 00 00 00 CC
```

<< back to index

Command	Code hex	Code decimal
Request Bill Id	0x9D	157

Implemented on
SMART PAYOUT

Description

This command will return the ascii bill data for a requested channel.

The return data is formatted as a 7 character identification code [C] [C] [V] [V] [V] [V] [I]

CC = Standard 2 letter country code e.g. EU for the euro.

VVVV = Bill value in terms of the country scaling factor

I = Issue code. Starts at A and progresses B, C, D, E.

The command takes 1 byte data which represents the bill channel of the ID required.

Packet examples

In this example we require the id data of channel 2.

```
Host transmit:  28  01  01  9D  02  37
  Slave Reply:  01  07  28  00  45  55  30  30  31  30  41  34
        ascii:                   E   U   0   0   1   0   A
```

If the channel is not supported, all zeros are returned.

```
Host transmit:  28  01  01  9D  06  33
  Slave Reply:  01  07  28  00  30  30  30  30  30  30  30  80
        ascii:                   0   0   0   0   0   0   0
```

<< back to index

Command	Code hex	Code decimal
Request Country Scaling Factor	0x9C	156

Implemented on
SMART PAYOUT

Description

A request to return the scaling factor and decimal place position for the given country code. The command data contains two byte ascii country code.

For a supported country code then 3 value bytes are returned.

bytes 0 and 1 are the value multiplier (scaling factor).

Byte 2 is the number of decimal places.

In this example scaling factor = 100, decimal places = 2. So a channel value of 100 would be a real value of 100 * 100/100 = 100.00

Packet examples

In this example, we request scaling factor for EURO (EU) returns 100

Host transmit: **28 02 01 9C 45 55 9F**

Slave Reply: **01 03 28 00 64 00 02 6E**

<< back to index

Command	Code hex	Code decimal
Request Bill Position	0x9B	155

Implemented on
SMART PAYOUT

Description

Use this command for obtaining the inhibit mask position of a given currency code. Two data bytes are returned.
This command data is 2 byte ascii code of the country required.

The inhibit mask based on the command currency code is returned.

Packet examples

In this example, a validator has euro currency on channel1 1,2 and 3 only.

```
Host transmit:  28 02 01 9B 45 55 A0
  Slave Reply:  01 02 28 00 07 00 CE
```

<< back to index

Command	Code hex	Code decimal
Route Bill	0x9A	154

Implemented on
SMART PAYOUT

Description

The host command to decide a destination for the bill held in escrow.
Command has 1 data byte containing a route code:

0 = Return escrow bill

1= send to stack

255 = extend escrow hold time.

Packet examples

This example is a command to return an escrow bill.

Host transmit:	**28 01 01 9A 00 3C**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Modify Bill Operating Mode	0x99	153

Implemented on
SMART PAYOUT

Description

Command data byte is bit field. Bit 0 is not used. Bit 1 is 0 to disable escrow, 1 to enable.

Packet examples

This example shows command to set escrow on the device.

Host transmit:	**28 01 01 99 02 3B**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Request Bill Operating Mode	0x98	152

Implemented on
SMART PAYOUT

Description

This command returns the status of the bill operating mode register.

It is configured as bit field. bit 0 is not used, bit 1 is the escrow function.

If the bit is set, the function is used, 0 = not used.

Packet examples

In this example escrow mode is set on the device.

```
Host transmit:  28 00 01 98 3F
  Slave Reply:  01 01 28 00 02 D4
```

<< back to index

Command	Code hex	Code decimal
Request Currency Revision	0x91	145

Implemented on
SMART PAYOUT

Description

This command returns an 8 byte ascii array identifying the dataset version on the device.

Packet examples

In this example the ascii for EUR02604 dataset is returned.

```
Host transmit:  28  00  01  91  46
  Slave Reply:  01  08  28  00  45  55  52  30  32  36  30  34  E7
        ascii:                   E   U   R   0   2   6   0   4
```

<< back to index

Command	Code hex	Code decimal
Switch Encryption Code	0x89	137

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The host can change the current BNV encryption key to a new value using this command. The command is encrypted with the old key and the response by the device encrypted with the old key.

The new key will then take effect but will only persist in memory until it is either changed again or the device is reset. Use the Store
Encryption Code command to persistently store this new code.

Packet examples

Example to set new key to 987654. Data bytes are packed so byte 0 = 89h (137 dec), byte 1 = 67h (103 dec), byte 2 = 45h (69 dec)

```
Host transmit:  28 03 01 89 89 67 45 16
  Slave Reply:  01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Store Encryption Code	0x88	136

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command stores the current BNV encryption key into non-volatile memory.

Packet examples

Host transmit:	**28 00 01 88 4F**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Read Barcode Data	0x81	129

Implemented on
SMART PAYOUT

Description

The host command to read data about a barcode that has been in escrow or stacked. If no barcode has been read, the response will be an ACK without any data bytes

Barcode Data is valid until a new note insertion is detected.

Packet examples

ACK with no data. No valid barcode data is stored.

```
Host transmit:  28 00 01 81 56
  Slave Reply:  01 00 28 00 D7
```

An example response of 16 digit barcode data 1234567812345678.

```
Host transmit:  28 00 01 81 56
  Slave Reply:  01 10 28 00 31 32 33 34 35 36 37 38 31 32 33 34 35 36 37 38 7F
       ascii:               1  2  3  4  5  6  7  8  1  2  3  4  5  6  7  8
```

<< back to index

Command	Code hex	Code decimal
Request Encryption Support	0x6F	111

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return data about the device encryption configuration to allow host machine installation and configuration. This command will respond even if the device has BNV encryption set and the command is sent unencrypted.

The commands data bytes act as a validation signature.

Response Data:

[Protocol level] [Command level] [Protocol key size] [Command key size] [Command block size] [Trusted mode] [BNV2 | BNV1] [BNV4 | BNV3] [BNV6 | BNV5] [DES1] [DES2] [DES3] [DES4] [DES5] [DES6] [DES7] [
DES8]

The validation data (0xAA 0x55 0x00 0x00 0x55 0xAA) bytes are sent to ensure that the command can not be confused if sent as an encrypted packet from another command.

Packet examples

<< back to index

Command	Code hex	Code decimal
Switch Des Key	0x6E	110

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to change the current DES key. The old key and new key are interleaved and the two blocks of 8 byte data are encrypted with the old key.

The device then decrypts the data and checks that the old keys match. If so then the swap is made an ACK is sent to the host and the new key stored in persistent memory.

If the keys do not match then no reply is sent.

This command can also be sent for key verification. If the host sets the new and old keys to the same value then the device will reply with an ACK if the key is correct or no reply if the key sent is not correct.

Packet examples

<< back to index

Command	Code hex	Code decimal
Request Encrypted Status	0x6D	109

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

To guard against ACK/NACK hijack frauds this command may be used to determine the status of the device.

The command may be sent after a payout or if a NACK was received to a payout command to verify the
status of the device.

The host send 3 challenge bytes, which are then embedded in the reply from the device to verify correct peripheral response.

Host Response data 16 bytes (DES encrypted):

[CRC Low][Challenge 1][Event count] [Last payout Request 0][Last payout Request 1][Last payout Request 2] [Last payout Request 3] [Last payout Amount 0][Last Amount Request 1][Last Amount Request 2] [Last payout Amount 3] [Challenge 2][Random 1][Random 2][Challenge 3][CRC High]

Packet examples

<< back to index

Command	Code hex	Code decimal
Get Inhibit Peripheral Device Value	0x35	53

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command returns the current status of the Master Inhibit value from the device indicated by the peripheral code.

Command has 8 data bytes. Byte 0 = periferal code, 0 = coin mech, 1 = coin feeder. Bytes 1-4 are the coin value, bytes 5-7 are the acsii country code.

ACK response with 1 data byte for success. Data byte shows state of coin inhibit 0 = disabled, 1 = enabled.

NACK with fail code for unsuccessful command. This example shows failure for currency mis-match.

code	fail reason
no return	Command not implemented
0	No device detected
1	Device out of service
2	Device currency miss-match

Packet examples

Example show enabled response to EUR 0.20

```
Host transmit:  28 08 01 35 01 14 00 00 00 45 55 52 99
  Slave Reply:  01 01 28 00 01 D5
```

<< back to index

Command	Code hex	Code decimal
Get Cashbox Operation Data	0x34	52

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command allows the host to obtain individual levels of each coin denomination emptied to the cashbox after a Smart empty operation.

The device responds with an ACK and an array of data: byte 0 - number of denominations, then 9 bytes for each of the denominations giving 2 bytes for the level of coins emptied, 4 bytes for the coin value and 3 bytes for the ascii country code. There the follows 4 bytes giving a count of unknown coins emptied.

Packet examples

This example shows that a hopper with 3 denominations of EUR coins was emptied. There were 10 x 0.20 cent coins, 50 x 1.00 coins and 20 x 2.00 coins. 5 unknown coins were also emptied.

```
Host transmit:  28  00  01  34  A3
  Slave Reply:  01  1C  28  00  03  0A  00  14  00  00  00  45  55  52  32  00  64  00  00  00  45  55  52  14
                00  C8  00  00  00  45  55  52  64
      ascii:                        .   .   .   .   .   .   .   E   U   R   2   .   d   .   .   .   E   U   R   .
                .   .   .   .   .   E   U   R
```

<< back to index

Command	Code hex	Code decimal
Smart Empty	0x33	51

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command will generate events during the emptying process and give values of the current amount emptied in response to Get Status commands.

This is encrypted level command and the example is shown as at a unencrypted level. Details of the encryption levels are not shown here.

Packet examples

Host transmit:	**28 00 01 33 A4**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Set Inhibit Peripheral Device Value	0x32	50

Implemented on
SMART HOPPER, SMART SYSTEM

Description

A command to enable or disable a coin value for the attached coin paying-in mechanism or coin feeder. The peripheral device type is address by the device code byte. If the command is not successfully executed, then a NACK will be returned with one data byte giving the reason failure. This value is stored in volatile ram and will be set to disabled state after a reset.

Command has 9 data bytes: Byte 0 - Peripheral code, 0 = coin mech, 1= coin feeder. Byte 1 enable state, 1 = enable, 0
= disable. Bytes 2-5 - 4 byte coin value. Bytes 6-8 the ascii country code.

NACK with fail code for unsuccessful command. This example shows failure for currency mis-match.

code	fail reason
no return	Command not implemented
0	No device detected
1	Device out of service
2	Device currency miss-match

Packet examples

This example is to enable coin value EUR 0.10 on a coin feeder unit.

Host transmit:	**28 0A 01 32 00 01 01 0A 00 00 00 45 55 52 A3**
Slave Reply:	**01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Get Peripheral Device Master Inhibit	0x31	49

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command returns the master inhibit status of a connected peripheral to the Smart Hopper.
The host sends 1 data parameter. 0 = coin mech, 1 = coin feeder.

For an error response, the device gives a error data byte with NAK.

data byte	function
non return	Command not implemented
0	Device not connected
1	Device out of service
2	Device currency miss-match

Packet examples

This example shows a request for the master inhibit of a coin feeder. The device response ACK and a data byte giving the status of the master inhibit. In this case 1 for enabled.

Host transmit:	**03 01 01 31 01 C9**
Slave Reply:	**01 01 03 00 01 FA**

Error NAK example showing the currency of the coin mech and hopper do not match.

Host transmit:	**03 01 01 31 00 CA**
Slave Reply:	**01 01 03 05 02 F4**

<< back to index

Command	Code hex	Code decimal
Set Peripheral Device Master Inhibit	0x30	48

Implemented on
SMART HOPPER, SMART SYSTEM

Description

A command to globally enable or disable the attached coin paying-in mechanism or coin feeder operations. The peripheral device type is address by the device code byte. If the command is not successfully executed, then a NACK will be returned with one data byte giving the reason failure. This value is stored in volatile ram and will be set to disabled state after a reset.

Command has two data bytes. Data 0 is the peripheral code, 0 = coin mech, 1 = coin feeder. Data 1 is the inhibit command. It is a bit field, bit 0 - inhibit status, 0 = device inhibited, 1 = device enabled. Bits 1 - 7 are not used.

For an error response, the device gives a error data byte with NAK.

data byte	function
non return	Command not implemented
0	Device not connected
1	Device out of service
2	Device currency miss-match

Packet examples

Example shows master inhibit set to enable device on a coin feeder unit.

```
Host transmit:  03 02 01 30 01 01 C8
  Slave Reply:  01 00 03 00 FC
```

NAK response with error code for failure. Example shows master inhibit set fail on coin feeder due to device out of service.

```
Host transmit:  03 02 01 30 01 01 C8
  Slave Reply:  01 01 03 05 01 F5
```

<< back to index

Command	Code hex	Code decimal
Request Status (cur)	0x2F	47

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the status of the device and the progress of the current requested operation. After issuing any action commands this command should be used to track the status.

See Event Table for list of possible event responses

Packet examples

Example response data for Dispensing event value EUR 5.30.

```
Host transmit:  03  00  01  2F  CD
  Slave Reply:  01  09  03  00  01  01  12  02  00  00  45  55  52  F1
        ascii:                  .   .   .   .   .   .   E   U   R
```

<< back to index

Command	Code hex	Code decimal
Get Device Setup (cur)	0x2E	46

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the setup of the device, the number of different types of coin/note and the value and currency of each coin/note that the device can handle. The length of the returned data will be 1+(n*7) bytes long, where n is the number of notes/coins that can be used.

Tabel showing set-up response:

byte	**function**	**size**
0	number of denominations in the device	1
1	denomination 0 value	4
5	denomination 0 country code	3
8	denomination 1 value	4
12	denomination 1 code	3
15	continued 7 bytes value and code for each denomination	...

Packet examples

This example shows a response for a system with 3 denominations. EUR 0.20, EUR 0.50 and GBP 1.00

```
Host transmit:  28  00  01  2E  A9
 Slave Reply:   01  16  28  00  03  14  00  00  00  45  55  52  32  00  00  00  45  55  52  64  00  00  00  2F
                2A  32  B1
       ascii:                   .   .   .   .   .   E   U   R   2   .   .   .   E   U   R   d   .   .   .   /
                *   2
```

<< back to index

Command	Code hex	Code decimal
Float By Denomination (cur)	0x2D	45

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a selected amount of denominations to be left in the device. All other notes/coins will be routed to the cashbox.

The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte. Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command.

Non-security levels just return an ACK.

ACK response is given for sucessfully accepted commands with security bytes plus1 byte event counter. Value is 0 at reset then increments on each successful command. After value 255 it wraps to 1.

Table showing command data format (security bytes not described here).

byte	function	size
0	number of denominations in request	1
1	number of denomination 0 to leave in float	2
3	value of denomination 0	4
7	country code of denomination 0	3
10	number of denomination 1 to leave in float	2
12	value of denomination 1	4
16	country code of denomination 1	3
19	repeat 9 byte block for each subsequent denomination	...

For non-successful command attempts, NAK is given with a data byte showing failure reason:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Payout By Denomination (cur)	0x2C	44

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be paid from the device specifying the quantity of denominations required. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte.

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command. Non-security levels just return an ACK.

Table showing command data format (security bytes not described here).

byte	function	size
0	number of denominations in request	1
1	number of denomination 0 to pay	2
3	value of denomination 0	4
7	country code of denomination 0	3
10	number of denomination 1 to pay	2
12	value of denomination 1	4
16	country code of denomination 1	3
19	repeat 9 byte block for each subsequent denomination	...

For non-successful command attempts, NAK is given with a data byte showing failure reason:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

This example is a un-secure request to pay 3 x 0.10 EUR, 7 x 1.00 EUR and 12 x 0.02 EUR.

```
Host transmit:  03 1D 01 2C 00 03 03 00 0A 00 00 00 45 55 53 07 00 64 00 00 00 45 55 52
                0C 00 02 00 00 00 45 55 52 65
  Slave Reply:  01 00 03 00 FC
        ascii:
```

<< back to index

Command	Code hex	Code decimal
Set Denomination Amount (cur)	0x2B	43

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command will add the number of coins specified in Count to the internal coin counter for the value specified in Value. If the count specified is Zero then the counter will be reset. It is not possible to set the absolute value to anything except zero in a single command.

This command should be used each time the coin acceptor routes a coin into the hopper, or when the hopper has coins manually added. This command is invalid for the Smart Payout as the notes are automatically added to the counter by the note validator.

Command data format:

byte	function	size
0	denomination value	4
4	level to add	2
6	denomination country code	3

Packet examples

ACK for success

Host transmit: **28 00 01 2B AC**

Slave Reply: **01 00 28 00 D7**

NACK for fail due to denomination/country not existing on system.

Host transmit: **28 00 01 2B AC**

Slave Reply: **01 00 28 05 D2**

<< back to index

Command	Code hex	Code decimal
Get Denomination Amount (cur)	0x2A	42

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the count of the number of coins/notes of the value specified in the command data.

byte	function	size
0	denonomination value	4
4	denomination country code	3

Packet examples

Example showing response of 10 for request of EUR 5.00

```
Host transmit:  03 07 01 2A F4 01 00 00 45 55 52 EA
  Slave Reply:  01 02 03 00 0A 00 F0
```

<< back to index

Command	Code hex	Code decimal
Get Minimum Payout (cur)	0x29	41

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the value of the minimum payout that is possible with the coins/notes that are currently in the device. This is effectively the value of the lowest coin/note in the device.

Command data is 3 data bytes. This contains the currency code for the minimum request. The device responds with 4 byte value of min payout.

Packet examples

Example asking for Min payout of EURO giving response EUR 5.00

```
Host transmit:  28 03 01 29 45 55 52 BF
  Slave Reply:  01 04 28 00 F4 01 00 00 DE
```

<< back to index

Command	Code hex	Code decimal
Float Amount (cur)	0x28	40

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be left in the device, paying out all excess monies into the device cashbox. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte.

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command.

Non-security levels just return an ACK.

Command data format (no security bytes shown):

byte	function	size
0	minimum payout remaining	4
4	float value	4
8	country code	3

Returns NAK with error code for request failures:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

This example is a request to float to 20.00 EUR with a min payout of 5.00 EUR

```
Host transmit:  28 0B 01 28 F4 01 00 00 D0 07 00 00 45 55 52 EC
  Slave Reply:  01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Get Routing (cur)	0x26	38

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the route setting for a particular value denomination. 0 for values routed to be stored for payout, 1 for values to be routed to the cashbox.

Command data format:

byte	function	size
0	value	4
4	country code	3

Packet examples

Example shows route on device has been set to cashbox for 0.50 euro coin

```
Host transmit:  03 07 01 26 32 00 00 00 45 55 00 03
  Slave Reply:  01 01 03 00 01 FA
```

<< back to index

Command	Code hex	Code decimal
Set Routing (cur)	0x25	37

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A command to control the route of a denomination entered in to the device. Money can either be stored in the device available for payout or sent to an external cashbox.

For routes to cashbox - In the case of the Smart Payout, notes entered will be routed straight to the cashbox; with the Smart Hopper/Smart System coins will be routed to the cashbox as they are detected by the hopper discrimination system.

Command data format:

byte	function	size
0	route (0 = payout, 1= cashbox)	1
1	value	4
5	country code	3

Packet examples

This example shows a command to set 5.00 EUR note to route to cashbox.

```
Host transmit:  28 08 01 25 01 F4 01 00 00 45 55 52 C8
  Slave Reply:  01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Set Bezel Mode	0x23	35

Implemented on
SMART PAYOUT

Description

A command to set the colour mix mode of the Smart Payout BNV Bezel.

Command data format:

byte	function	size
0	Bezel type (see table below)	1
1	Red PWM	1
2	GreenPWM	1
3	Blue PWM	1
4	Store mode (0= RAM, 1= EEPROM)	1
5	not used - set to 0	3

Bezel types:

Byte value	Description
0	Standard (steady colour when enabled)
1	Flashing bezel when enabled
2	Bezel colour when disabled (all PWM values to 255 will turn bezel off when disabled)

Packet examples

This example sets the bezel color to RED to be stored in the EEPROM.

Host transmit: **28 09 01 23 00 FF 00 00 01 00 00 00 00 AB**

Slave Reply: **01 00 28 00 D7**

<< back to index

Command	Code hex	Code decimal
Run Unit Calibration	0x22	34

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A command to set the host to run its self-calibration function. This is done in response to a calibration fault event type 7 and is used so that the host can optimise its power spread by determining when the device motors will activate.

This command will only function if the Host Calibration option is enabled.

Packet examples

```
Host transmit:  28 00 01 22 B5
 Slave Reply:   01 00 28 00 D7
```

<< back to index

Command	Code hex	Code decimal
Float By Denomination	0x21	33

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a selected amount of denominations to be left in the device. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte.

Successful commands sent with the security levels set return an event count byte.

This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command. Non-security levels just return an ACK.

Table showing command data format (security bytes not described here).

byte	function	size
0	number of denominations in request	1
1	number of denomination 0 to leave in float	2
3	value of denomination 0	4
7	number of denomination 1 to leave in float	2
9	value of denomination 1	4
13	repeat 6 byte block for each subsequent denomination	...

For non-successful command attempts, NAK is given with a data byte showing failure reason:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Payout By Denomination	0x20	32

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be paid from the device specifying the quantity of denominations required. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte.

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command. Non-security levels just return an ACK.

Table showing command data format (security bytes not described here).

byte	function	size
0	number of denominations in request	1
1	number of denomination 0 to pay	2
3	value of denomination 0	4
7	number of denomination 1 to pay	2
9	value of denomination 1	4
13	repeat 6 byte block for each subsequent denomination	...

For non-successful command attempts, NAK is given with a data byte showing failure reason:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Get Payout Options	0x1F	31

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return two bytes giving the current status of the device options. REG 0 is transmitted first.

Response table format:

function	action	default	device
REG 0			
bit 0	Pay mode 0 = free pay, 1 = High value split	1	Smart Hopper, Smart System
bit 1	Level check 0 = disabled, 1 = enabled	1	Smart Hopper, Smart System
bit 2	Motor speed 0 = low, 1 = high	1	Smart Hopper, Smart System
bit 3	Not used set to 0	0	
bit 4	Payout algorithm normal = 0, high speed split = 1	0	
bit 5	Unknown coin route Cashbox = 0, Payout = 1	0	
bit 6	Not used set to 0	0	
bit 7	Not used set to 0	0	
REG 1			
bit 0	Enable Note In Bezel Hold message. 1 = enabled, 0 = disabled.	0	Smart Payout only
bit 1	Enable Note Stored at Startup message. 1 = enabled	0	Smart Payout only
bit 2	Not used set to 0	0	
bit 3	Not used set to 0	0	
bit 4	Not used set to 0	0	
bit 5	Not used set to 0	0	
bit 6	Not used set to 0	0	
bit 7	Not used set to 0	0	

Packet examples

<< back to index

Command	Code hex	Code decimal
Set Payout Options	0x1E	30

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The host can set run-time options for the SMART Hopper, SMART Payout or SMART System device.

Command has two date bytes with configuration:

function	action	default	device
REG 0			
bit 0	Pay mode 0 = free pay, 1 = High value split	1	Smart Hopper, Smart System
bit 1	Level check 0 = disabled, 1 = enabled	1	Smart Hopper, Smart System
bit 2	Motor speed 0 = low, 1 = high	1	Smart Hopper, Smart System
bit 3	Not used set to 0	0	
bit 4	Payout algorithm normal = 0, high speed split = 1	0	
bit 5	Unknown coin route Cashbox = 0, Payout = 1	0	
bit 6	Not used set to 0	0	
bit 7	Not used set to 0	0	
REG 1			
bit 0	Enable Note In Bezel Hold message. 1 = enabled, 0 = disabled.	0	Smart Payout only
bit 1	Enable Note Stored at Startup message. 1 = enabled	0	Smart Payout only
bit 2	Not used set to 0	0	
bit 3	Not used set to 0	0	
bit 4	Not used set to 0	0	
bit 5	Not used set to 0	0	
bit 6	Not used set to 0	0	
bit 7	Not used set to 0	0	

Packet examples

<< back to index

Command	Code hex	Code decimal
Request Status	0x1D	29

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the status of the device and the progress of the current requested operation. After issuing any action commands this command should be used to track the status.

See Event Tables for list of possible event responses

Packet examples

<< back to index

Command	Code hex	Code decimal
Get Device Setup	0x1C	28

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the setup of the device, the number of different types of coin/note and the value and currency of each coin/note that the device can handle. The length of the returned data will be 4+(n*4) bytes long, where n is the number of notes/coins that can be used.

Tabel showing set-up response:

byte	**function**	**size**
0	country code of dataset	3
1	number of denominations	1
2	denomination 0 value	4
6	denomination 1 value	4
10	continued 4 bytes value and code for each denomination	...

Packet examples

<< back to index

Command	Code hex	Code decimal
Set Denomination Amount	0x1B	27

Implemented on
SMART HOPPER, SMART SYSTEM

Description

This command will add the number of coins specified in Count to the internal coin counter for the value specified in Value. If the count specified is Zero then the counter will be reset. It is not possible to set the absolute value to anything except zero in a single command.

This command should be used each time the coin acceptor routes a coin into the hopper, or when the hopper has coins manually added. This command is invalid for the Smart Payout as the notes are automatically added to the counter by the note validator.

Command data format:

byte	function	size
0	denomination value	4
4	level to add	2

Packet examples

This example shows 20 EUR 0.50 coins being added to the system.

```
Host transmit:  03 06 01 1B 20 00 00 00 14 00 A7
  Slave Reply:  01 00 03 00 FC
```

<< back to index

Command	Code hex	Code decimal
Get Denomination Amount	0x1A	26

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the count of the number of coins/notes of the value specified in the command data.

byte	function	size
0	denonomination value	4

Packet examples

An example command requesting the number of EUR 5.00 notes stored in the device. Returns 12 notes

```
Host transmit:  28 04 01 1A F4 01 00 00 C4
  Slave Reply:  01 02 28 00 0C 00 C9
```

<< back to index

Command	Code hex	Code decimal
Get Minimum Payout	0x19	25

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command will return the value of the minimum payout that is possible with the coins/notes that are currently in the device. This is effectively the value of the lowest coin/note in the device.

Packet examples

Example response showing device has a minimum payout available of EUR 0.20

```
Host transmit:  03 00 01 19 E3
  Slave Reply:  01 04 03 00 14 00 00 00 E4
```

<< back to index

Command	Code hex	Code decimal
Empty	0x18	24

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request the device to empty all of its store of monies into the external cashbox forcollection. No values of cashbox payout are given during the empty procedure and after the device has emptied, the denomination counters will all be set to zero. The format of the command depends on the security level setting of the device.

This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its
data, the [Command level] byte. Successful commands sent with the security levels set return an event count byte. This
byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty
command

. Non-security levels just return an ACK.

Packet examples

<< back to index

Command	Code hex	Code decimal
Float Amount	0x17	23

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command allows the host to request a value amount to be left in the device, paying out all excess monies into the device cashbox. The format of the command depends on the security level setting of the device. This may be obtained using the Request Encryption Support (header 111) command, which returns as part of its data, the [Command level] byte.

Successful commands sent with the security levels set return an event count byte. This byte is value 0 at reset and then wraps from 255 to 1. It increments on every successful payout, float or empty command.

Non-security levels just return an ACK.

Command data format (no security bytes shown):

byte	function	size
0	minimum payout remaining	4
4	float value	4

Returns NAK with error code for request failures:

error code	error reason
1	Not enough value in device
2	Cannot pay this exact amount
3	Device busy
4	Device disabled
5	Device lid/path open
6	Device jam
7	Calibration error
8	Fraud detected
9	Device disconnected

Packet examples

<< back to index

Command	Code hex	Code decimal
Get Routing	0x15	21

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

This command returns the route setting for a particular value denomination. 0 for values routed to be stored for payout, 1 for values to be routed to the cashbox.

Command data format:

byte	function	size
0	value	4

Packet examples

Example command for requesting route of EUR 5.00. Returns cashbox

Host transmit: **28 04 01 15 F4 01 00 00 C9**
Slave Reply: **01 01 28 00 01 D5**

<< back to index

Command	Code hex	Code decimal
Request Hopper Status	0xA6	166

Implemented on
SMART SYSTEM

Description

This command will return the event number and the payout status and the payout requested.

Packet examples

<< back to index

Command	Code hex	Code decimal
Payout Amount By Denomination	0x24	36

Implemented on
SMART SYSTEM

Description

This command is similar to 'Payout amount' command but selecting the coin denominations used in two monetary values with the format:

Value1(4bytes), Number of denomination in request (1 byte), Value of coin, Repeat for each denomination

Value2(4bytes), Number of denomination in request (1 byte), Value of coin, Repeat for each denomination

Packet examples

<< back to index

Command	Code hex	Code decimal
Set Cashbox Payout Limits	0x36	54

Implemented on
SMART SYSTEM

Description

Allow the host to specify a maximum level of coins, by denomination, to be left in the hopper.

During any payout operation, if there are coins in the hopper in excess of the set levels, when they are encountered on the conveyor belt they will be sent to the cashbox (beneath the hopper).

This means that over time (and multiple payout operations) any excess coins will be sent to the cashbox and the desired level will be achieved.

It effectively allows the hopper to do the 'floating' for the host machine i.e. it is an auto float mechanism.

NB: If a coin route is changed from cashbox to payout and then back to cashbox then the level for this coin will be reset to 0 (any of the coins will then be sent to cashbox).

Command format.

byte	function	size
0	The number of individual requests	1
1	The level limit to set	2
3	The denomination value	4
7	The denomination country code (3 byte ASCII)	3
...	**Repeat above block for each denomination required**	**...**

Packet examples

<< back to index

Command	Code hex	Code decimal
Halt	0x40	64

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Stops the current payout operation. Any bills/coins in the process of paying-out will be paid and no further payouts given.

A HALTED event is generated giving the value of payout at halt.

Packet examples

Halt command

Host transmit:	**07 00 01 40 B8**
Slave Reply:	**01 00 07 F0 08**

<< back to index

Event	Code hex	Code decimal
Dispensing	0x01	1

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Give the current value being dispensed

Type	Data size (bytes)	Repeat
Pay-out	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value dispensing

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**yes**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data dispensing is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

Response showing dispensing value of 12.50

```
Host transmit:  01 00 28 1D BA
  Slave Reply:  01 05 28 00 01 E2 04 00 00 EB
```

<< back to index

Event	Code hex	Code decimal
Note Read	0x14	20

Implemented on
SMART PAYOUT

Description

The device is currently reading a banknote. The event data gives the value of the note. A zero value indicates that the note value is not yet determined.

Type	Data size (bytes)	Repeat
Status	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value of the note being read.

Type	Data size (bytes)	Repeat
Status	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data gives the value of the note being read is given as:

byte	function	size
0	value	4
4	country code	3

Packet examples

<< back to index

Event	Code hex	Code decimal
Idle	0x00	0

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Additional infomation

Given when the device is ready

Packet examples

Response showing device idle

Host transmit:	**01 00 28 1D BA**
Slave Reply:	**01 01 28 00 00 D6**

<< back to index

Event	Code hex	Code decimal
Dispensed	0x02	2

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

Gives the complete value dispensed for this transaction.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value dispensed

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data dispensed is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Coins Low	0x03	3

Implemented on
SMART HOPPER, SMART SYSTEM

Description

Given when the system detects that the level of coins in the systen has reached a low level and the ystem requires refilling.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Empty	0x04	4

Implemented on
SMART HOPPER, SMART SYSTEM

Description

Type	Data size (bytes)	Repeat
Pay-out	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Halted	0x06	6

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The devie has been halted in response to a request from the host.

Type	Data size (bytes)	Repeat
Status	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value at halt.

Type	Data size (bytes)	Repeat
Status	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data at halt is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Floating	0x07	7

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The value currently floated in the device.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value floated up to the status request.

Type	Data size (bytes)	Repeat
Status	**variable**	**yes**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data floated up to the status request is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Floated	0x08	8

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The final amount floated by the device.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value floated

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data floated is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Timeout	0x09	9

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device detected no more valid notes/coins available for payout during a payout or float operation.

The value paid up until the time-out point is given in the event data.

Type	Data size (bytes)	Repeat
Error	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value paid/floated up until the time-out.

Type	Data size (bytes)	Repeat
Error	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data gives value paid/floated up until the time-out::

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Cashbox Paid	0x0C	12

Implemented on
SMART HOPPER, SMART SYSTEM

Description

Gives the value of coins that have been paid out via the cashbox route (because they were designed 'route cashbox' during a payout operation.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value paid to the cashbox.

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data paid to cashbox is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Coin Credit	0x0D	13

Implemented on
SMART HOPPER, SMART SYSTEM

Description

A coin has been detected as added to the system via the attached coin mechanism.

Type	Data size (bytes)	Repeat
Pay-in	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value added

Type	Data size (bytes)	Repeat
Pay-in	**variable**	**yes**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data of coin added is given as:

byte	function	size
0	coin value	4
4	coin country	3

Packet examples

<< back to index

Event	Code hex	Code decimal
Emptying	0x0E	14

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device is currently performing a request to empty itself of coins/notes.

Type	Data size (bytes)	Repeat
Pay-out	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Emptied	0x0F	15

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device has competed it's empty operation.

Type	Data size (bytes)	Repeat
Pay-out	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Fraud Attempt	0x10	16

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device has detect an attempt to fraud the system for false payments.

The amount paid/floated before the fraud attempt is given in the even data.

Type	Data size (bytes)	Repeat
Fraud	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value up unitl the fraud was detected.

Type	Data size (bytes)	Repeat
Fraud	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data up until the fraud was detected is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Disabled	0x11	17

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device is disabled for normal pay-in/payout operations.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Note Stored	0x12	18

Implemented on
SMART PAYOUT

Description

A banknote has been stored in the payout device.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Slave Reset	0x13	19

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

A device has undergone a power reset process.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

Example showing reset response

Host transmit:	**01 00 28 1D BA**
Slave Reply:	**01 01 28 00 13 C3**

<< back to index

Event	Code hex	Code decimal
Note Credit	0x15	21

Implemented on
SMART PAYOUT

Description

A banknote has moved to the safe credit position. It's value is given in the event data.

Type	Data size (bytes)	Repeat
Pay-in	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value of the note to credit.

Type	Data size (bytes)	Repeat
Pay-in	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data gives the value of the note to credit is given as:

byte	function	size
0	value	4
4	country code	3

Packet examples

<< back to index

Event	Code hex	Code decimal
Note Rejecting	0x16	22

Implemented on
SMART PAYOUT

Description

The device is currently rejecting an invalid bill back to the user.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Rejected	0x17	23

Implemented on
SMART PAYOUT

Description

The device has rejected an invalid bill back to the user.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Stacking	0x18	24

Implemented on
SMART PAYOUT

Description

The device is moving a valid bill to the stacker area.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Stacked	0x19	25

Implemented on
SMART PAYOUT

Description

A valid bill has been moved to the device stacker and the stacking process has completed.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Note Path Jam	0x1A	26

Implemented on
SMART PAYOUT

Description

The device has detected that a bill has jammed during reading and is stuck the note path.

Type	Data size (bytes)	Repeat
Error	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Note Stack Jam	0x1B	27

Implemented on
SMART PAYOUT

Description

The device has detected that a note has jammed during it's transport to the stacker.

Type	Data size (bytes)	Repeat
Error	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Bill From Front At Start	0x1C	28

Implemented on
SMART PAYOUT

Description

At power up, a bill that was in the note path at power down has been rejected from the front of the device back to the user.

If the value is know, this is given in the event data.

Type	Data size (bytes)	Repeat
Status	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support:

0x4-data byes give the bill value. If this is not know then 0 value is given.

Type	Data size (bytes)	Repeat
Status	**7**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support:

The bill value is given as 4 byte value and 3 byte ascii countrt code if known. If not then all zeros are given.

Packet examples

<< back to index

Event	Code hex	Code decimal
Bill Stacked At Start	0x1D	29

Implemented on
SMART PAYOUT

Description

At power-up a bill that was in the stacking mechanism during power down was stacked. If the value was know, this is given in the event data.

Type	Data size (bytes)	Repeat
Status	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support:

0x4-data byes give the bill value. If this is not know then 0 value is given.

Type	Data size (bytes)	Repeat
Status	**7**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support:

The bill value is given as 4 byte value and 3 byte ascii countrt code if known. If not then all zeros are given.

Packet examples

<< back to index

Event	Code hex	Code decimal
Cashbox Full	0x1E	30

Implemented on
SMART PAYOUT

Description

The device cashbox stacker has been filled to capacity and requires emptying before the device can be re-enabled for operation.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Cashbox Removed	0x1F	31

Implemented on
SMART PAYOUT

Description

The device cashbox has been removed and must be replaced before the device can be re-enabled for operation.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Cashbox Replaced	0x20	32

Implemented on
SMART PAYOUT

Description

The device cashbox has been replaced and the devie is available for normal operation.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Lid Open	0x21	33

Implemented on
SMART HOPPER

Description

The device cashbox lid is open and the device is disbaled from operation.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Lid Closed	0x22	34

Implemented on
SMART HOPPER

Description

The device cashbox lid has been closed.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Calibration Fault	0x24	36

Implemented on
SMART HOPPER, SMART SYSTEM

Description

The device has detected an error in it's sensor calibration system.

The event data byte gives the code of the error concerned.

Type	Data size (bytes)	Repeat
Error	**1**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
Attached Mech Jam	0x25	37

Implemented on
SMART HOPPER, SMART SYSTEM

Description

The optional coin mechanism attached to this device has been detected as jammed.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Attached Mech Open	0x26	38

Implemented on
SMART HOPPER, SMART SYSTEM

Description

The optional coin mechanism attached to this device has been detected as open.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Smart Emptying	0x27	39

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The devie is undergoing it's smart emptying process. All the coins/bill are being paid into the cashbox and their accumulated values are given in the event data.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value empited so far.

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data showing the amount emptied so far is given as is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Smart Emptied	0x28	40

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

The device has completed it's smart empty process and the values emptied to the casbx are given in the event data.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value emptied

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data emptied is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Barcode Escrow	0x34	52

Implemented on
SMART PAYOUT

Description

A barcode ticket has been successfully scanned and is held in the escrow position.

The barcode data is retrieved using the Read Barcode Data command and the ticket can either be rejected or accepted in to the device as requried.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Barcode Stacked	0x35	53

Implemented on
SMART PAYOUT

Description

A valid barcode ticket has been moved to and sucessfully stacked in the stacker unit.

Type	Data size (bytes)	Repeat
Status	**0**	**no**

Packet examples

<< back to index

Event	Code hex	Code decimal
Multiple Value Added	0x36	54

Implemented on
SMART SYSTEM

Description

An accumulated value added to the system since the last Request staus command.

Type	Data size (bytes)	Repeat
Pay-in	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes giving the acculated value paid-in since the last request.

Type	Data size (bytes)	Repeat
Pay-in	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data of accumulated value paid-in since the last request is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Peripheral Error	0x37	55

Implemented on
SMART SYSTEM

Description

The system periferal device has generated an error.

Type	Data size (bytes)	Repeat
Error	**2**	**no**

Additional infomation

The event data gives two bytes the peripheral code (0 for coin mech, 1 for coin feeder) and the error code.

Packet examples

<< back to index

Event	Code hex	Code decimal
Peripheral Device Disabled	0x38	56

Implemented on
SMART SYSTEM

Description

The peripheral device is currently disabled for operation.

Type	Data size (bytes)	Repeat
Status	**1**	**yes**

Additional infomation

The event data byte gives the peripheral (0 = coin mech, 1= coin feeder)

Packet examples

Example showing coin feeder device disabled

Host transmit:	**01 00 28 1D BA**
Slave Reply:	**01 02 28 00 38 01 9C**

<< back to index

Event	Code hex	Code decimal
Bill Held In Bezel	0x39	57

Implemented on
SMART PAYOUT

Description

A dispensed banknote is currently held in the bezel

This event is only given if the option is enabled in Set Payout Options command.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**yes**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

4 data bytes give the value held.

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**yes**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data for value held is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Value Pay-in	0x3A	58

Implemented on
SMART SYSTEM

Description

Payin active.

Packet examples

<< back to index

Event	Code hex	Code decimal
Incomplete Payout	0x0A	10

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

At start-up a discrepancy between the last paid amount and last requested amount was detected.

Type	Data size (bytes)	Repeat
Pay-out	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support.

8 data bytes give the value dispensed and then the value requested

Type	Data size (bytes)	Repeat
Pay-out	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support.

Curency data showing values dispensed and requested are given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value dispensed	4
6	value requested	4
10	country code	3
9	Repeat values and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Incomplete Float	0x0B	11

Implemented on
SMART HOPPER, SMART PAYOUT, SMART SYSTEM

Description

At start-up a discrepancy between the last floated amount and last requested amount was detected.

Packet examples

<< back to index

Event	Code hex	Code decimal
Device Full	0x3B	59

Implemented on
SMART SYSTEM

Description

Coin/cashbox full level reached.

Packet examples

<< back to index

Event	Code hex	Code decimal
Bill Stored At Startup	0x3C	60

Implemented on
SMART PAYOUT

Description

During power-up this event is given if a bill was being stored in the Smart Payout during power down and no credit event had been given before power loss. It contains the value of the note being stored.

This event must be enabled by setting Reg 1 Bit 1 of Set Payout Options. If this event is not enabled, it will report the note as stacked at startup to ensure the credit isn't lost. This event will never be in the first poll response, to ensure enabling it as a response to Slave Reset will always be valid.

Type	Data size (bytes)	Repeat
Status	**4**	**no**

Additional infomation

Response to Request Status (0x1D) command with no currency support:
0x4-data byes give the bill value. If this is not know then 0 value is given.

Type	Data size (bytes)	Repeat
Status	**7**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support:
The bill value is given as 4 byte value and 3 byte ascii countrt code if known.

Packet examples

<< back to index

Event	Code hex	Code decimal
Error During Payout	0x30	48

Implemented on
SMART PAYOUT

Description

Type	Data size (bytes)	Repeat
Error	**variable**	**no**

Additional infomation

Response to Request Status (0x2F) command with currency support. This is used instead of halted when the payout operation is cancelled automatically by the device.

Curency data at halt is given as:

byte	function	size
0	Event code	1
1	Number of currencies in event	1
2	value	4
6	country code	3
9	Repeat value and country code for each denomination in the event.	

Packet examples

<< back to index

Event	Code hex	Code decimal
Payout Jam Recovery	0x31	49

Implemented on
SMART PAYOUT

Description

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Additional infomation

Response to Request Status (0x2F) command with currency support. This event is given when the payout is performing a jam recovery operation.

Packet examples

<< back to index

Event	Code hex	Code decimal
Startup Initialisation Active	0x32	50

Implemented on
SMART PAYOUT

Description

This event is shown as a response to Request Status with Multi Currency support (0x2F). It shows that the unit is still performing startup actions and isn't ready to accept / dispense notes.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples

<< back to index

Event	Code hex	Code decimal
All Channels Inhibited	0x33	51

Implemented on
SMART PAYOUT

Description

This event is given as a response to Request Status with MultiCurrency Support(0x2F). It shows that all of the Note Channels have been inhibited.

Type	Data size (bytes)	Repeat
Status	**0**	**yes**

Packet examples