SECTION 5

NV11 MANUAL SET

SOFTWARE IMPLEMENTATION GUIDE

INTELLIGENCE IN VALIDATION

Innovative Technology
INTELLIGENCE IN VALIDATION

Bellis Technology
INTELLIGENCE IN VALIDATION

NV11 MANUAL SET – SECTION 5

5. SOFTWARE IMPLEMENTATION GUIDE

5.1 Communication Protocols

The NV11 validator can use two different communication protocols - SSP/eSSP and ccTalk.

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

Encrypted Smiley® Secure Protocol (eSSP) is an enhancement of SSP. eSSP uses the same 16 bit CRC checksums on all packets as SSP, but also uses a Diffie-Hellman key exchange to allow the host machine and validator to jointly establish a shared secret key over an insecure communications channel. The encryption algorithm used is AES with a 128-bit key; this provides a very high level of security.

The recommended communication protocol for the NV11 validator is eSSP, as this provides the highest level of data transfer security. A ccTalk interface protocol is also available.

For detailed information and the full protocol specifications please read the following documents, which can be downloaded from the Innovative Technology Ltd website (www.innovative-technology.co.uk):

- SSP Interface Specification (ITL Document number GA138)
- ccTalk Communications Protocol Specification [additional commands] (ITL Document number GA964)
- ITL Bank Note Reader ccTalk Specification (ITL Document number GA966)

Summaries of the NV11 validator socket connections for the supported interfaces are shown below:

WARNING!

Risk of unit damage

Do not make any connections to the interface socket pins marked '**Do not connect**' – making connections to these pins could cause severe damage to the unit.

Information

Encryption of data strongly recommended

It is recommended that all transactions with the NV11 validator be encrypted to prevent commands being recorded and replayed by an external device. If this is not possible, then other (mechanical) measures should be used to prevent physical bus tapping.

NV11 SSP Interface:

Pin	Name	Type	Description
1	TxD	Output	Serial data out (Tx)
2	Factory use only		**Do not connect**
3			
4			
5	RxD	Input	Serial data in (Rx)
6	Factory use only		**Do not connect**
7			
8			
9			
10			
11	USB D+	Data	USB Data +
12	USB D-	Data	USB Data -
13	USB Vcc	Power	USB +V (+5V)
14	Factory use only		**Do not connect**
15	V In	Power	+V
16	GND	Ground	GND

NV11 ccTalk Interface:

Pin	Name	Type	Description
1	TxD	Output	ccTalk data – must also be connected to pin 5
2	Factory use only		**Do not connect**
3			
4			
5	RxD	Input	ccTalk data – must also be connected to pin 1
6	Factory use only		**Do not connect**
7			
8			
9			
10			
11	USB D+	Data	USB Data +
12	USB D-	Data	USB Data -
13	USB Vcc	Power	USB +V (+5V)
14	Factory use only		**Do not connect**
15	V In	Power	+V
16	GND	Ground	GND

5.2 Configuration Card Programming

It is not possible to program the NV11 by the use of a configuration card as this method of programming is not yet implemented. Full details on programming the NV11 validator using software can be found in Section 3 of this manual set (ITL Software Support Guide).

5.3 SSP and eSSP

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

Encrypted Smiley® Secure Protocol (eSSP) is an enhancement of SSP. eSSP uses the same 16 bit CRC checksums on all packets as SSP, but also uses a Diffie-Hellman key exchange to allow the host machine and validator to jointly establish a shared secret key over an insecure communications channel. The encryption algorithm used is AES with a 128-bit key; this provides a very high level of security.

The encryption of the SSP protocol ensures superior protection and reliability of the data, which is transferred between validator and host machine. The encryption key is divided into two parts:

- The lower 64 bits are fixed and specified by the machine manufacturer allowing control of which devices are used in their machines.
- The higher 64 bits are securely negotiated by the slave and host at power up, ensuring each machine and each session are using different keys.

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 transmission with simple open collector drivers. Each SSP device of a particular type has a unique serial number; this number is used to validate each device in the direction of credit transfer before transactions can take place.

Information

200 ms command spacing

When communicating with the NV11 validator, poll commands should be sent **at least** 200 ms apart.

SSP Commands and Responses

a. Commands

Action	Command Code (Hex)	Command Set
Reset	0x01	Generic
Host Protocol Version	0x06	
Poll	0x07	
Get Serial Number	0x0C	
Synchronisation command	0x11	
Disable	0x09	
Enable	0x0A	
Program Firmware / currency	0x0B (Programming Type)	
Manufacturers Extension	0x30 (Command, Data)	
Set inhibits	0x02	Validator
Display On	0x03	
Display Off	0x04	
Set-up Request	0x05	
Reject	0x08	
Unit data	0x0D	
Channel Value data	0x0E	
Channel Security data	0x0F	
Channel Re-teach data	0x10	
Last Reject Code	0x17	
Hold	0x18	

Action	Command Code (Hex)	Command Set
Enable Protocol Version Events	0x19 (made obsolete in protocol version 6)	Validator
Get Bar Code Reader Configuration	0x23	
Set Bar Code Reader Configuration	0x24	
Get Bar Code Inhibit	0x25	
Set Bar Code Inhibit	0x26	
Get Bar Code Data	0x27	
Enable Payout Device *	0x5C, Options	Note Float
Disable Payout Device	0x5B	
Set Routing *	0x3B (route, value/channel (Country code))	
Get Routing *	0x3C (value/channel (Country code))	
Empty	0x3F	
Get Note Positions *	0x41	
Payout Note	0x42	
Stack Note	0x43	
Set Value Reporting Type *	0x45 (type)	
SMART empty	0x52	
Cashbox Payout Operation Data	0x53	

The commands marked with * will respond with the generic response 'Command cannot be processed' and an error code of 'Invalid Currency' if there are notes inside the Note Float module that do not match the dataset that is installed in the validator.

Notes:

Action	Comments
Reset:	Single byte command, causes the slave to reset
Host Protocol Version:	Dual byte command, the first byte is the command; the second byte is the version of the protocol that is implemented on the host.
Poll:	Single byte command, no action taken except to report latest events.
Get Serial Number:	Single byte command, used to request the slave serial number. Returns 4-byte long integer.
Sync:	Single byte command, which will reset the validator to expect the next sequence ID to be 0.
Disable:	Single byte command, the peripheral will switch to its disabled state, it will not execute any more commands or perform any actions until enabled, any poll commands will report disabled.
Enable:	Single byte command, the peripheral will return to service.
Manufacturers Extension:	This command allows the manufacturer of a peripheral to send commands specific to their unit
Enable Payout Device:	Two-byte command, the first byte is the command and the second enables several different options. Unused bits should be 0.
Disable Payout Device:	Single-byte command. All accepted notes will be routed to the stacker and payout commands will not be accepted.
Set Routing:	The first byte is the command. The second byte is the selected route, and the remaining bytes are the four-byte value or single byte channel number (depending on the reporting type selected) of the note that the route should be applied to. By default all note values are stacked. For protocol version 6, three extra country code bytes are sent
Empty:	Single byte command, this will cause all notes to be sent to the stacker for removal.
Get Note Positions:	Single byte command, this will return the number of notes in the Note Float module and the value in each position. The first note in the table is the first note that was paid into the Note Float module - the Note Float module is a LIFO system, so the note that is last in the table is the only one that is available to be paid out or moved into the stacker.

Action	Comments
Payout Note:	This is a single byte command. The Note Float module will pay out the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the payout is possible the Note Float will reply with the generic response 'OK'.
Stack Note:	This is a single byte command. The Note Float module will stack the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the stack operation is possible the Note Float will reply with the generic response 'OK'.
Set Value Reporting Type:	Two-byte command. This will set the method of reporting values of notes. There are two options, by a four-byte value of the note or by the channel number of the value from the banknote validator. If the channel number is used then the actual value must be determined using the data from the Validator command Unit Data. The default operation is by 4-byte value.
SMART empty:	Empties the Note Float module of notes, maintaining a count of value emptied. The note counters will be set to 0 after running this command. Use 'cashbox payout operation data' command to retrieve a breakdown of the denominations routed to the cashbox through this operation.
Cashbox Payout Operation Data:	This command can be sent at the end of a SMART Empty, float or dispense operation. Returns the amount emptied to cashbox from the Note Float in the last dispense, float or empty command. The quantity of denominations in the response is sent as a 2 byte little endian array; the note values as 4-byte little endian array and the country code as a 3-byte ASCII array. Each denomination in the dataset will be reported, even if 0 notes of that denomination are emptied.

b. Responses

Action	Command Code (Hex)	Command Set
OK	0xF0	Generic
Command not known	0xF2	
Wrong number of parameters	0xF3	
Parameter out of range	0xF4	
Command cannot be processed	0xF5	
Software Error	0xF6	
FAIL	0xF8	
Key Not Set	0xFA	
Slave Reset	0xF1	Validator
Read, n	0xEF, Channel Number	
Credit, n	0xEE, Channel Number	
Rejecting	0xED	
Rejected	0xEC	
Stacking	0xCC	
Stacked	0xEB	
Safe Jam	0xEA	
Unsafe Jam	0xE9	
Disabled	0xE8	
Fraud Attempt, n	0xE6, Channel Number	
Stacker Full	0xE7	
Note cleared from front at reset	0xE1, Channel Number	

Action	Command Code (Hex)	Command Set
Note cleared into cash box at reset	0xE2, Channel Number	Validator
Cash Box Removed	0xE3	
Cash Box Replaced	0xE4	
Bar Code Ticket Validated	0xE5	
Bar Code Ticket Acknowledge	0xD1	
Note Path Open	0xE0	
Channel Disable	0xB5	
Dispensing	0xDA, value/channel dispensing	Note Float
Dispensed	0xD2, value/channel dispensed	
Jammed	0xD5, value/channel dispensing	
Halted	0xD6	
Incomplete Payout	0xDD, value/channel dispensed, value/channel requested	
Emptying	0xC2	
Empty	0xC3	
Note Stored in Payout *	0xDB, value/channel of note	
Note Transferred to Stacker	0xC9, value/channel of note	
Payout Out of Service	0xC6	
Note Paid into Stacker at Power Up	0xCA, value/channel of note	
Note Paid into Store at Power Up	0xCB, value/channel of note	

Action	Command Code (Hex)	Command Set
Note Dispensed at Power Up	0xCD, value/channel of note	Note Float
Note Float Removed	0xC7	
Note Float Attached	0xC8	
Note in Bezel Hold	0xCE, value/channel of note	
Device Full	0xCF	
SMART Emptying	0xB3	
SMART Emptied	0xB4	
Channel Disable	0xB5	

* When enabled using option flag.

Notes:

Action	Comments
Command Not Known:	Returned when an invalid command is received by a peripheral.
Wrong Number Of Parameters:	A command was received by a peripheral, but an incorrect number of parameters were received.
Parameter Out Of Range:	One of the parameters sent with a command is out of range.
Command Cannot Be Processed:	A command sent could not be processed at that time.
Software Error:	Reported for errors in the execution of software e.g. Divide by zero. This may also be reported if there is a problem resulting from a failed remote firmware upgrade, in this case the firmware upgrade should be redone
Key Not Set:	The slave is in encrypted communication mode but the encryption keys have not been negotiated
Jammed:	Five-byte response that indicates that the validator is jammed; this is reported until it is un-jammed or reset. It will also become disabled.

Action	Comments
Dispensing:	Single-byte response indicating that a dispense operation is in progress. The four-byte value of the note or the single byte channel number is reported, depending on the reporting type set. This value will be 0 until the note has passed out of the Note Float module and into a payable position in the validator.
Dispensed:	Response that indicates when the payout has finished a dispense operation; The four-byte value of the note or the single byte channel number is reported, depending on the reporting type set.
Jammed:	Five byte response that indicates that the payout is jammed; this is reported until it is un-jammed or reset. It will also become disabled. The value or channel number of the note being dispensed is also reported.
Incomplete Payout:	This event is given when the payout starts up if a payout or float operation was in progress when the power was removed. The first four bytes after the event code are the value that was dispensed; the next four are the value that was originally requested.
Emptying:	This event is given while the payout is being emptied of notes into the cashbox by the EMPTY command.
Empty:	This event is given at the end of the empty process.
Note Stored in Payout:	This event is given when notes paid in to the payout system are routed to the payout store. For compatibility with the SMART Payout the value of the note is not reported. However if the Note Float is enabled with the option flag VALUE_ON_STORED set, then the value of the note will be reported (see enable payout device command).
Note Transferred to Stacker:	This event is given when a note has successfully been moved from the Note Float and stacked in the cash box. During the process the Stacking event will be given.
Note Float Removed:	This event is reported when the Note Float module is physically disconnected from the validator while the power is on.
Note Float Attached:	This event is reported when the Note Float module is physically attached to the validator while the power is on. The validator and Note Float module will then reset.
Note In Bezel Hold:	This event is reported when the Dispensing note is held in the bezel waiting for the user to remove it.

Action	Comments
Device Full:	This event is reported when the Note Float has reached its limit of stored notes. This event will be reported until a note is paid out or stacked.
Channel Disable:	Indicates all note channels have been inhibited and as such, the unit is disabled. Only reported if using protocol version 7 and above.

Example SSP Communications

Here is an example of the communication between host and slave. Both the typical commands from the host and responses from the validator are detailed.

Host	Slave	Comments
> SYNC	< OK	Synchronisation command
> SET_GENERATOR, *[64 bit prime number]*	< OK	Set the encryption key generator
> SET_MODULUS, *[64 bit prime number]*	< OK	Set the encryption key modulus
> REQUEST_KEY_EXCHANGE *[64 bit host intermediate key]*	< OK, *[64bit slave intermediate key]*	Host sends the host intermediate key, slave responds with the slave intermediate key. The encryption key is then calculated independently by both host and slave.
> GET_SERIAL	< OK < [*SERIAL NUMBER*]	NV11 Serial Number
> SETUP_REQUEST	< OK < [*SETUP INFORMATION*]	NV11 Setup
> SET_ROUTING, 01 14 00 00 00	< OK	Route notes of value 0020 to the NV11 Cashbox
> SET_INHIBIT > 07 > 00	< OK	Enable channels 1,2 and 3
> ENABLE	< OK	Enable NV11
> POLL	< OK < DISABLED	
> POLL	< OK	
> POLL	< OK < NOTE READ < 00	NV11 currently reading a note
> POLL	< OK < NOTE READ < 03	Note has been recognised as channel 3 (€20)
> HOLD	< OK	Hold the note in escrow
> POLL	< OK < STACKING	Stack the note
> POLL	< OK < CREDIT < 03 < STACKING < STACKED	Credit given for channel 3 (€20), note stacked
> POLL	< OK	

Full support is available from ITL and local support offices for implementing eSSP - this support includes libraries and example applications. When requesting this information, please specify your preferred language(s) and operating system.

5.4 ccTalk

This section should be read in conjunction with the full ccTalk specification, which can be downloaded from the internet (www.cctalk.org).

ccTalk is a serial communications protocol in widespread use throughout the money transaction industry. Peripherals such as coin acceptors, note validators and hoppers found in a diverse range of automatic payment equipment use ccTalk to communicate with the host controller.

The protocol uses an asynchronous transfer of character frames in a similar manner to RS232. The main difference is that it uses a single two-way communication data line for half-duplex communication rather than separate transmit and receives lines. It operates at TTL voltages and is 'multi-drop' (peripherals can be connected to a common bus and are logically separated by a device address) - each peripheral on the ccTalk bus must have a unique address.

Each communication sequence (a command or request for information) consists of 2 message packets structured in one of the formats detailed below. The first packet will go from the master device to the slave device and then a reply will be sent from the slave device to the master device.

Commands can have 3 primary formats:

- 8 Bit Checksum – No Encryption
- 16 Bit CRC – No Encryption
- 16 Bit CRC – BNV Encryption

As it is possible to use the ccTalk protocol without encryption, suitable physical security should be employed to protect the ccTalk bus.

Information

200 ms command spacing

When communicating with the NV11 validator, Read Buffered Bill events (command 159) should be sent **at least** 200 ms apart.

ccTalk Command Summary

Command	Header	Parameters	Example
Reset Device	001	None	ACK
Request Comms Revision	004	None	X.Y
Force Empty NV11 Store to Cash Box	019	SEL	EVENT
Request Expanded NV11 status	020	SEL	FLAG, EVENT, REMAIN, PAID, UNPAID, STORED, STORE, FLAG2
Clear Total Count	021	SEL	ACK / NAK
Pump RNG	022	R1-R8	ACK
Request Cipher Key	023	None	KEY1 – KEY8
Request Variable Set	024	SEL	KEY, MC
Modify Variable Set	025	SEL, KEY	ACK
Request Total Count	026	SEL	IN1-IN3, OUT1-OUT3, STR1-STR3
Enable Payout	027	KEY	ACK / NAK
Dispense NV11 Notes	028	SEL, SEC1-SEC8, BILL CNT	EVENT / NAK
Request NV11 Status	029	SEL	FLAG, EVENT, REMAIN, PAID, UNPAID, STORED, STORE
Emergency Stop	030	SEL, FUNC	REMAIN
Empty NV11 Store to Cash Box	031	SEL	EVENT
Modify Variable MC Set	032	SEL, MC	ACK / NAK
Request RC Version	033	SEL	VERSION
Request RC Count	034	SEL	COUNT
Modify RC Count	035	SEL, COUNT	ACK / NAK
Request Current Count	036	SEL	COUNT
Read Barcode Data	129	None	ACK
Store Encryption Code	136	None	ACK
Switch Encryption Code	137	3 bytes Encryption key	ACK
Request Currency Revision	145	None or Country Code (2 digit)	'GBP02113'
Operate Bi-directional Motors	146	None	ACK
Stacker Cycle	147	None	ACK
Request Bill Operating Mode	152	None	3

Command	Header	Parameters	Example
Modify Bill Operating Table	153	Escrow & Stacker	ACK
Route Bill	154	0/1	ACK/254
Request Bill Position	155	Country Code (2 digit)	00000111 00000000
Request Country Scaling	156	Country Code (2 digit)	100
Request Bill ID	157	None	'GB0010A'
Read Buffered Bill Events	159	None	10000000000
Request Address Mode	169	None	1
Request Base Year	170	None	2006
Request Build Code	192	None	161209
Request Last Mod Date	195	None	00
Calculate ROM Checksum	197	None	4 byte checksum
Request Option Flags	213	None	3 (stacker & escrow)
Request Data Storage Av.	216	None	00000
Enter Pin	218	Pin1, Pin2, Pin3, Pin4	ACK
Enter New Pin	219	Pin1, Pin2, Pin3, Pin4	ACK
Request Accept Count	225	None	3
Request Insertion Count	226	None	7
Request Master Inhibit	227	None	1
Set Master Inhibit	228	Bit Mask	ACK
Request Inhibits	230	None	Inhibit Low, Inhibit High
Set Inhibits	231	Channels	ACK
Perform Self Check	232	None	0
Request Software Version	241	None	XX.YY
Request Serial Number	242	None	3 byte serial number
Request Product Code	244	None	'NV11'
Request Equipment Category	245	None	'Bill Validator'

Command	Header	Parameters	Example
Request manufacturer ID	246	None	'ITL'
Request Polling Priority	249	None	200
Simple Poll	254	None	ACK

Monetary Values

Values are represented as 32 bit unsigned integers (4 bytes) and in the lowest value of currency. For example:

€50.00 would be 0x00001388

When sending or receiving a value the least significant byte is sent first. So in this example [0x88] [0x13] [0x00] [0x00] will be sent.

Each type of note is identified by its value and represented using the standard format outlined above. As an example, the values for Euro notes are:

Note (€)	Hex value	Data to Send
5.00	0x000001F4	[0xF4] [0x01] [0x00] [0x00]
10.00	0x000003E8	[0xE8] [0x03] [0x00] [0x00]
20.00	0x000007D0	[0xD0] [0x07] [0x00] [0x00]
50.00	0x00001388	[0x88] [0x13] [0x00] [0x00]
100.00	0x00002710	[0x10] [0x27] [0x00] [0x00]
200.00	0x00004E20	[0x20] [0x4E] [0x00] [0x00]
500.00	0x0000C350	[0x50] [0xC3] [0x00] [0x00]

Communications Format

ccTalk communication is carried out using the following settings:

Baud rate= 9600bps, Data= 8 bit, No parity, Stop bit= 1bit

Command Format from Host to NV11

[Dst] [Len] [CRC (LSB)] [Header] [Data1] … [DataN] [CRC (MSB)]

Field	Description
Dst	Destination Address (for a note validator, 40 [0x28])
Len	Number of data bytes in the message
Header	Command
Data	Data attached to Header, specified by Len (1-N)
CRC	[Dst] + [Len] + [Header] + [Data1] + … + calculated value of CRC16 against Data N

Response Format from the NV11 to Host

ACK message:

[Dst] [00] [CRC (LSB)] [00] [CRC (MSB)]

NAK message:

[Dst] [00] [CRC (LSB)] [05] [CRC (MSB)]

BUSY message:

[Dst] [00] [CRC (LSB)] [06] [CRC (MSB)]

Response:

[Dst] [Len] [CRC (LSB)] [00] [Data1] … [DataN] [CRC (MSB)]

Field	Description
Dst	Destination Address (for a note validator, 40 [0x28])
Len	Number of data bytes in the message
Header	Reply message [00]
Data	Data attached to Header, specified by Len (1-N)
CRC	[Dst] + [Len] + [Header] + [Data1] + … + calculated value of CRC16 against Data N. If encryption setting is enabled, encryption is carried out against: [CRC (LSB)] + [Header] + [Data1] + … + [DataN] + [CRC (MSB)]

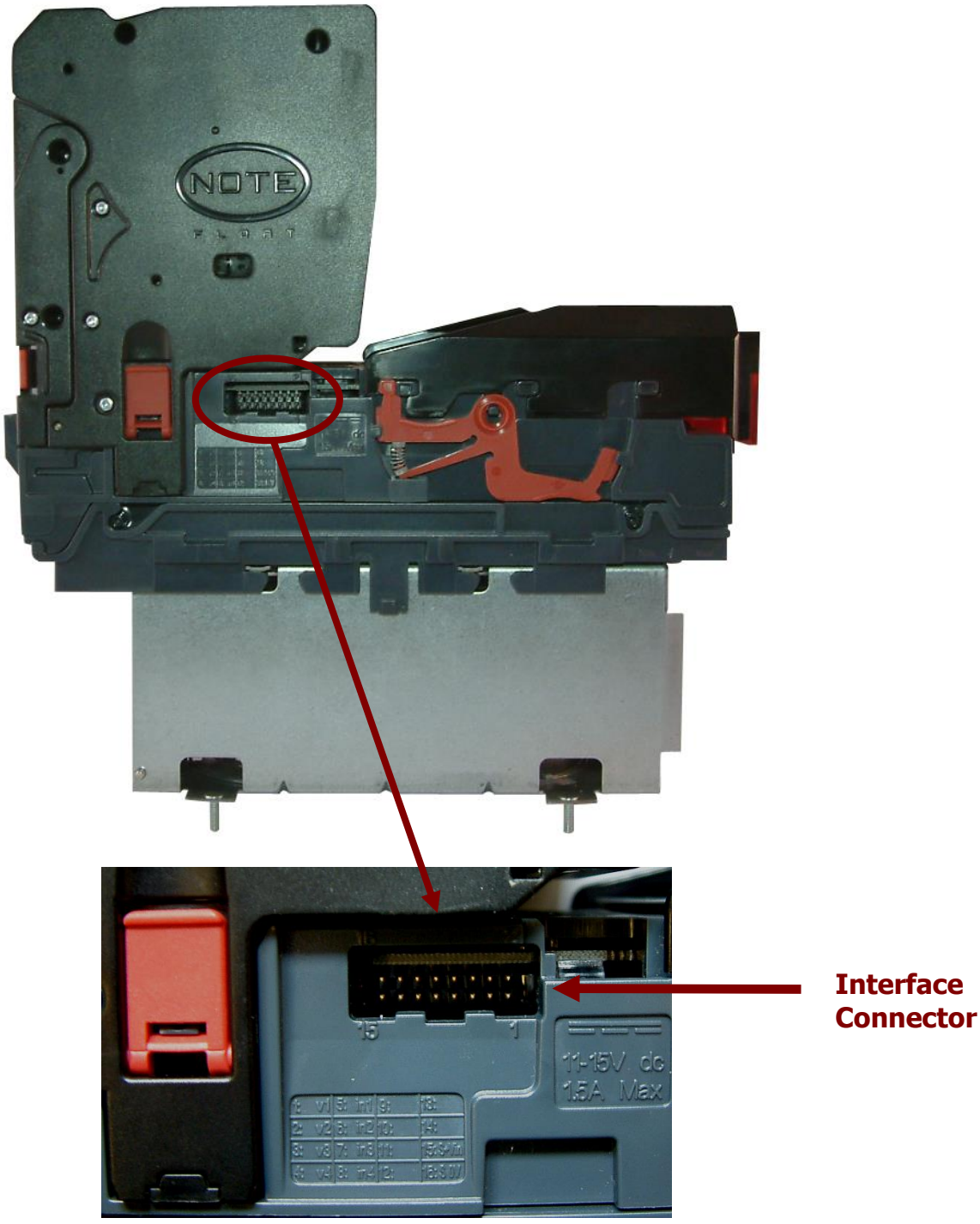
5.5 SSP Escrow Function

To hold a note in the escrow position when using SSP, the POLL command should be replaced with the HOLD command after NOTE READ > 0 for as long as the note is to be held in escrow.

A POLL command will then accept the note; the REJECT command will return the note to the customer

5.6 Connection Options

The NV11 validator has a single connector that is used to allow interfacing and programming:

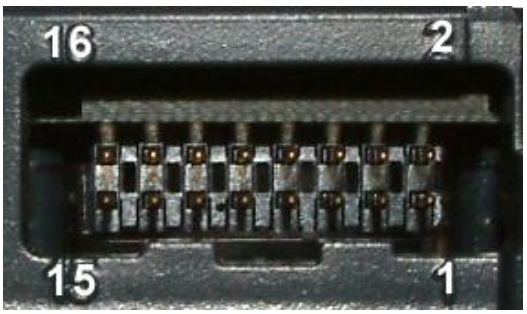


Information

Power always required regardless of connection type.

Power is always required on pins 15 and 16 of the 16 way connector.

The connector is a 16 pin socket used to interface the NV11 to the host machine. The pin numbering of the socket is shown below, as well as an overview of the socket connections:



Pin	Description
1	Serial Data Out (Tx)
5	Serial Data In (Rx)
11	USB Data +
12	USB Data -
13	USB Power (+5V)
15	+ V
16	0V / Ground Connection

To use a USB connection with the NV11, a USB cable fitted with a 16 way connector on one end (ITL Part Number CN00392) should be used. The CN00392 cable fits into the 16 way connector and allows high speed programming and serial communications when used in SSP and ccTalk modes.

Further details of the cable needed to interface and program the NV11 validator can be found in Section 4 of this manual set (subsection 4.9). When using the USB connection, power must be supplied to the NV11 using the CN00392 cable.

MAIN HEADQUARTERS

Innovative Technology Ltd
Derker Street – Oldham – England - OL1 4EQ
Tel: +44 161 626 9999 Fax: +44 161 620 2090
E-mail: support@innovative-technology.co.uk
Web site: www.innovative-technology.co.uk

Innovative Technology
INTELLIGENCE IN VALIDATION

Bellis Technology
INTELLIGENCE IN VALIDATION

AUSTRALIA
support@innovative-technology.com.au

BRAZIL
suporte@bellis-technology.com.br

CHINA
support@innovative-technology.co.uk

GERMANY
supportDE@innovative-technology.eu

ITALY
supportIT@innovative-technology.eu

SPAIN
supportES@innovative-technology.eu

UNITED KINGDOM
support@innovative-technology.co.uk

REST OF THE WORLD
support@innovative-technology.co.uk