SECTION 5

NV9 USB MANUAL SET

SOFTWARE IMPLEMENTATION GUIDE

INTELLIGENCE IN VALIDATION

SaXXot Deutschland GmbH, Zeppelinstrasse 73, DE 81669 München
Tel.: +49 (0)894141446-00 Fax: +49 (089) 4141446-75, info@saxxot.de - www.saxxot.de

NV9 USB MANUAL SET – SECTION 5

5. SOFTWARE IMPLEMENTATION GUIDE

5.1 Communication Protocols

The NV9 USB validator can use several different communication protocols, including eSSP, SIO, ccTalk, MDB, Parallel, Binary and Pulse. Use of the MDB protocol requires the use of an external IF5 interface unit.

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 NV9 USB 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 GA00138)
- ITL Bank Note Reader ccTalk Specification (ITL Document number GA00966)

Summaries of the NV9 USB 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 NV9 USB 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.

NV9 USB SSP Interface:

Pin	Name	Type	Description
1	Vend 1	Output	Serial data out (Tx)
2	Factory use only		**Do not connect**
3			
4			
5	Inhibit 1	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

NV9 USB ccTalk Interface:

Pin	Name	Type	Description
1	Vend 1	Output	Serial data – must also be connected to pin 5
2	Factory use only		**Do not connect**
3			
4			
5	Inhibit 1	Input	Serial 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

NV9 USB SIO Interface:

Pin	Name	Type	Description
1	Vend 1	Output	Serial data
2	Factory use only		**Do not connect**
3			
4			
5	Inhibit 1	Input	Serial data
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
When operating with this interface, the host machine does not echo messages back to the validator, and the NV9 USB does not operate in true RS232 mode (only TTL level).			

NV9 USB Pulse Interface:

Pin	Name	Type	Description
1	Vend 1	Output	Credit pulse stream output
2	Factory use only		**Do not connect**
3			
4			
5	Inhibit 1	Input	Inhibit Channel 1 by holding this pin HIGH
6	Inhibit 2	Input	Inhibit Channel 2 by holding this pin HIGH
7	Inhibit 3	Input	Inhibit Channel 3 by holding this pin HIGH
8	Inhibit 4	Input	Inhibit Channel 4 by holding this pin HIGH
9	Busy	Output	Busy signal – output is pulled low when the validator is busy
10	Escrow	Input	Enable Escrow function by holding this pin LOW
11	Factory use only		**Do not connect**
12			
13			
14			
15	V In	Power	+V
16	GND	Ground	GND

When operating in Pulse mode the NV9 USB outputs a number of pulses on Vend 1. The number of pulses for each channel is different and set to default values within the dataset. The number of pulses and the pulse duration can be modified using the Bank Note Validator Currency Manager Software, and a maximum of 16 channels can be used.

NV9 USB Multi Drop Bus (MDB) Interface:

MDB is a serial bus interface commonly used in electrically controlled vending machines. This is a 9600 Baud Master – Slave system where the NV9 USB validator is a slave to master controller.

To use the NV9 USB with MDB protocol, an **IF5** external interface is required. The IF5 regulates the power supply and opto-isolates the communication lines. The NV9 USB validator supports the MDB Protocol Version 1, Level 1.

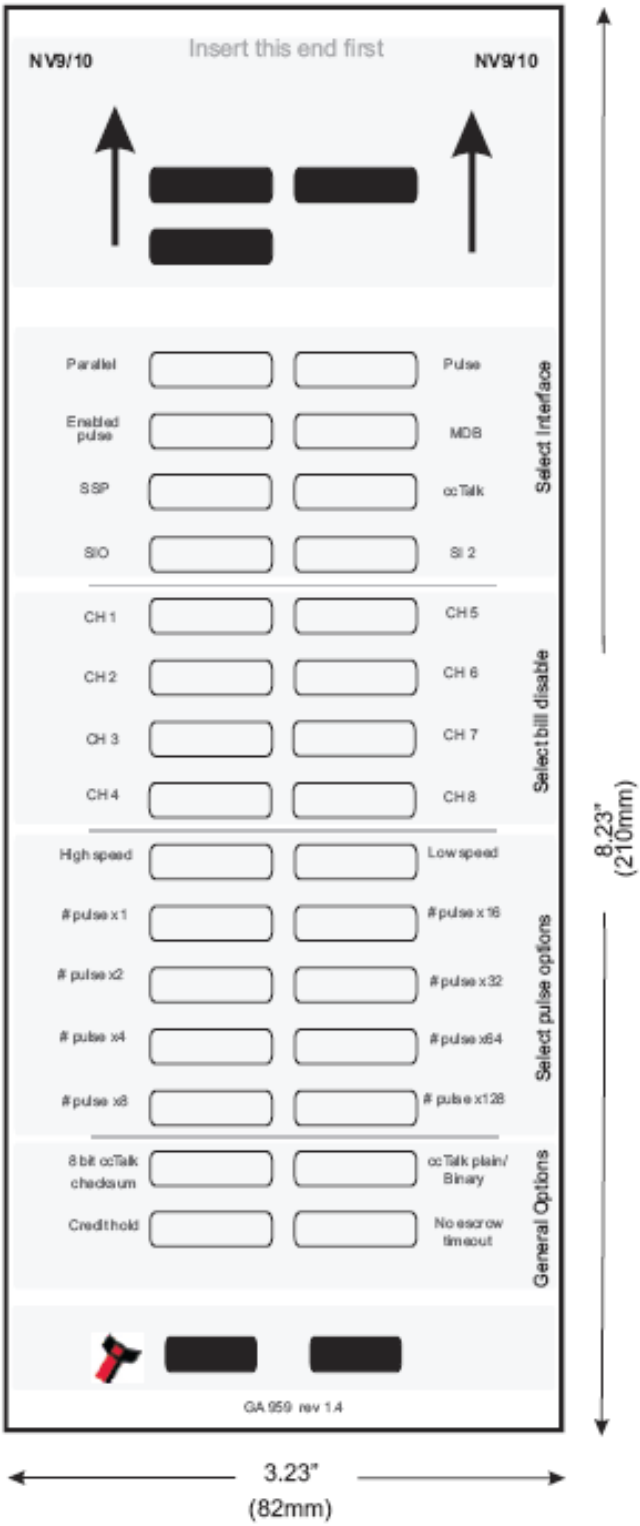
NV9 USB Parallel Interface:

Pin	Name	Type	Description
1	Vend 1	Output	Channel 1 credit, 100ms active low pulse
2	Vend 2	Output	Channel 2 credit, 100ms active low pulse
3	Vend 3	Output	Channel 3 credit, 100ms active low pulse
4	Vend 4	Output	Channel 4 credit, 100ms active low pulse
5	Inhibit 1	Input	Inhibit Channel 1 by holding this pin HIGH
6	Inhibit 2	Input	Inhibit Channel 2 by holding this pin HIGH
7	Inhibit 3	Input	Inhibit Channel 3 by holding this pin HIGH
8	Inhibit 4	Input	Inhibit Channel 4 by holding this pin HIGH
9	Busy	Output	Busy signal – output is pulled low when the validator is busy
10	Escrow	Input	Enable Escrow function by holding this pin LOW
11	Factory use only		**Do not connect**
12			
13			
14			
15	V In	Power	+V
16	GND	Ground	GND

When operating in Parallel mode the NV9 USB will issue a 100ms active LOW pulse on the relevant vend line, and a maximum of 4 channels can be used. There is also the option to use a binary output where the NV9 USB will output a binary pattern on vend lines 1 – 4. Binary mode can be set as an option using a configuration card or with the Bank Note Validator Currency Manager Software.

5.2 Configuration Card Programming

Please consult ITL technical document GA959 for further information on configuration card programming – the GA959 document includes a printable template for the configuration card and this can be downloaded from the Support section of the ITL website – **the sample shown here should not be used for programming as it is not to scale**.



Configuration Card - instructions for use:

1. Cut card around the outline - check the measurements are as printed. Make sure that 'Page scaling' is set to 'None' in your print options to ensure the correct size.

2. Fill in sections as required. Take care to fill in the sections correctly, keep inside the lines and fill boxes fully as shown here:

3. Power-up the validator and wait until it resets.

4. Press the configuration button once to enter programming mode (the bezel LEDs should flash at 1 second intervals).

5. Insert the card into the validator face up and in the direction indicated by the arrows.

6. The configuration card will be ejected and if the configuration was good the bezel LEDs will flash at a fast rate while programming takes place. After completion of programming the validator will reset.

Information

Check print settings.

Make sure that 'Page scaling' is set to 'None' in your print options to ensure the correct size when printing the configuration card.

If an error has occurred, the card will be rejected and the bezel LEDs will flash slowly a number of times to indicate the cause of the error:

Number of flashes	Indicated error
2	Invalid card read – card entered wrong way around, misread or wrong card version used
3	No interface selection was detected on the card
4	Multiple interface selections detected
5	Invalid interface detected – the selected interface is not available for this validator
6	Selected interface is not compatible with this validator version
7	Pulse configuration error – selected pulse options are invalid
8	ccTalk configuration error – the selected ccTalk options are invalid (ccTalk 8 bit checksum not allowed without ccTalk plain)
9	Low power mode not available for this validator version

WARNING!

Risk of unit damage

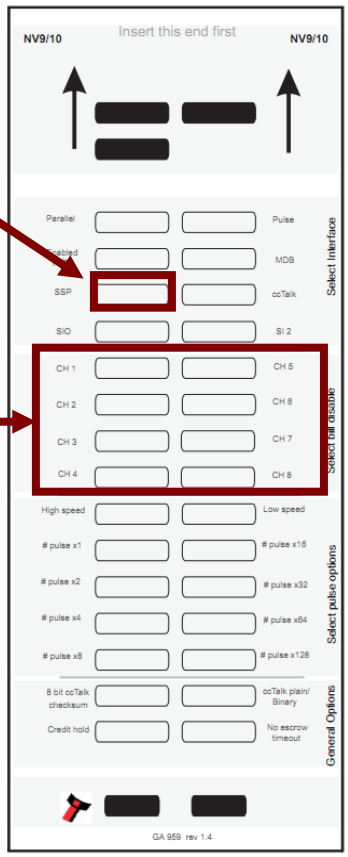
When in programming mode, do not turn off the power before the operation is complete as this will make the unit unusable.

There are different options available to use with the validator, depending on which interface is selected. Full details on programming the NV9 USB Validator using software can be found in Section 3 of this manual set (ITL Software Support Guide) – details of programming the various interfaces by use of configuration card are detailed on the next pages.

SSP configuration card options:

When using the configuration card to set SSP mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes



ccTalk configuration card options:

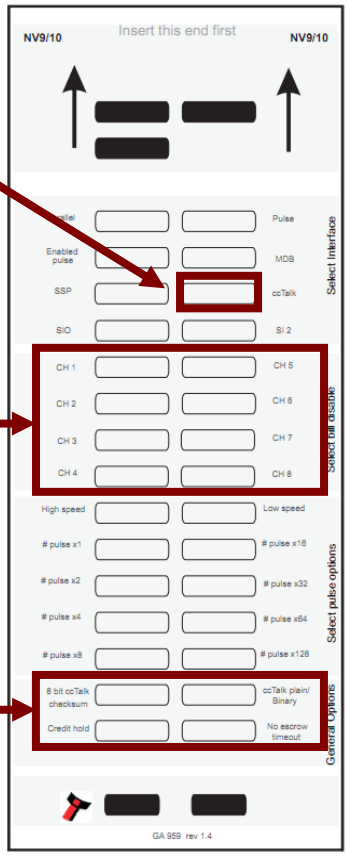
When using the configuration card to set ccTalk mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes

ccTalk plain – fill in this box to use un-encrypted ccTalk

ccTalk 8 bit checksum – if ccTalk plain is selected, leave this box blank for 16 bit CRC checksum. Fill in this box to use simple 8 bit checksum

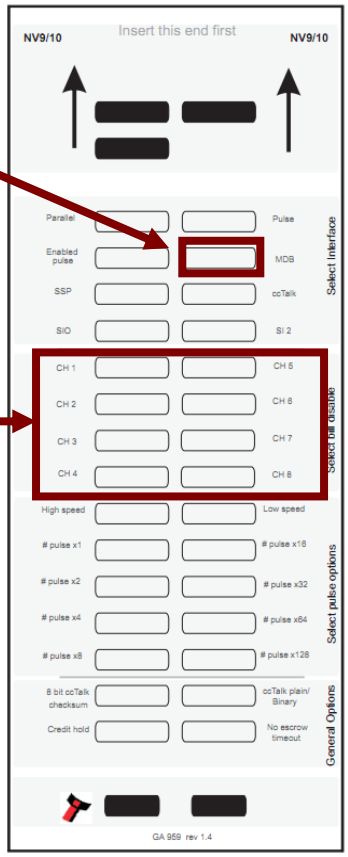
No escrow timeout – fill in this box to disable the escrow timeout



MDB configuration card options:

When using the configuration card to set MDB mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes



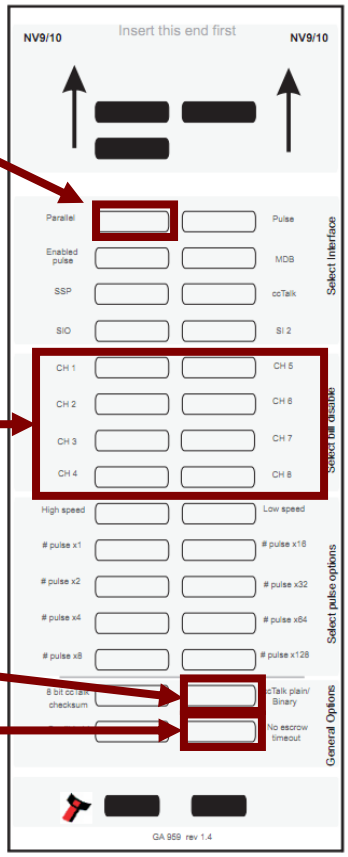
Parallel configuration card options:

When using the configuration card to set Parallel mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes

Binary – fill in this box to enable binary output mode

No escrow timeout – fill in this box to disable the escrow timeout



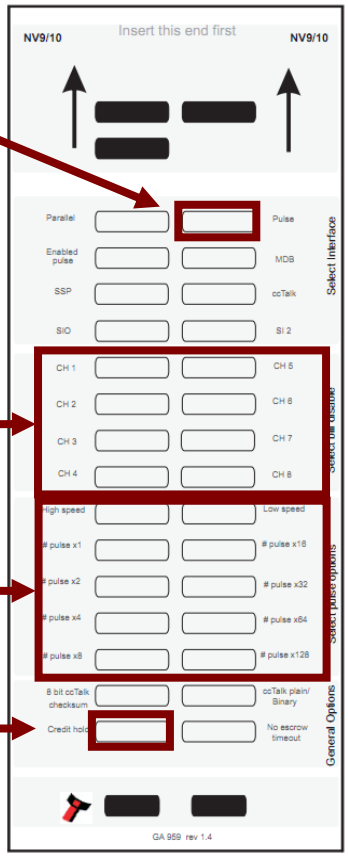
Pulse configuration card options:

When using the configuration card to set Pulse mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes

Pulse settings – set the required pulse widths and pulse multiplier

Credit hold – fill in this box to enable the credit hold function



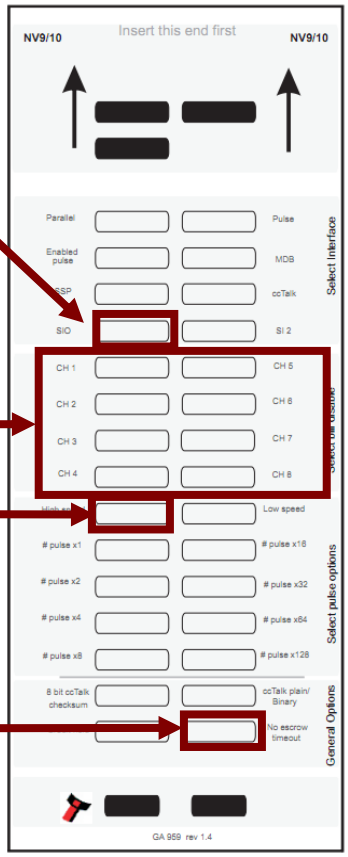
SIO configuration card options:

When using the configuration card to set SIO mode, the following options are available:

Note inhibits – fill in the boxes to inhibit notes

High speed – fill in this box to use high speed (9600 baud) data transfer rates output mode

No escrow timeout – fill in this box to disable the escrow timeout



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 serial number can be checked by the host on start up and receipt of a credit event to ensure that the device has not been changed.

Information

200 ms command spacing

When communicating with the NV9 USB 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)	
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	

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.

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, Error Code	
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
Note path open	0xE0	
Channel Disable	0xB5	

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 – this will return a corresponding error code.
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.

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*]	NV9 USB Serial Number
> SETUP_REQUEST	< OK < [*SETUP INFORMATION*]	NV9 USB Setup
> SET_ROUTING, 01 14 00 00 00	< OK	Route notes of value 0020 to the NV9 USB Cashbox
> SET_INHIBIT > 07 > 00	< OK	Enable channels 1,2 and 3
> ENABLE	< OK	Enable NV9 USB
> POLL	< OK < DISABLED	
> POLL	< OK	
> POLL	< OK < NOTE READ < 00	NV9 USB 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 NV9 USB 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
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
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

Command	Header	Parameters	Example
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	'NV9 USB'
Request Equipment Category	245	None	'Bill Validator'
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.

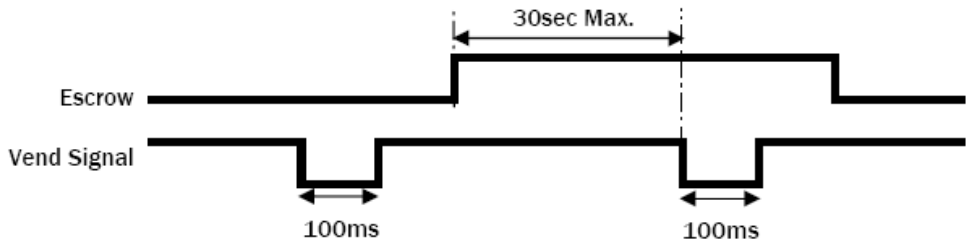
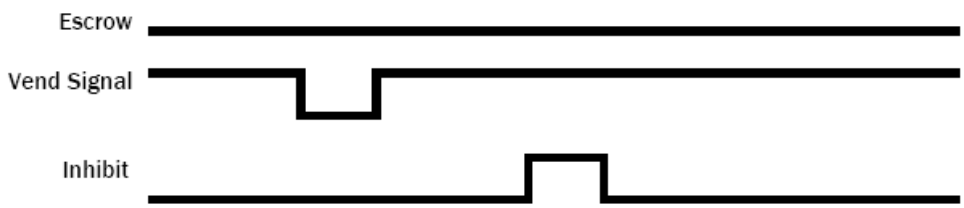
Each type of note is identified by its value and represented using the standard format outlined above. As an example, the values for Euro notes are:

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]

5.5 Escrow Control

The NV9 USB has a single note escrow facility (pin 10) used in Parallel, Pulse and Binary modes. This allows the Validator to hold onto the note once accepted, and only stack the note into the cash box when the host machine confirms that the vend operation has been completed.

If no confirmation of vend is received then the note will be returned to the customer after 30 seconds (see the escrow timing diagrams below):

	Do not wait more than 30 seconds for the 2nd vend confirmation signal
	Inhibit rejection can be at any time during the 30 second decision period after completion of the vend

If the host machine itself aborts the transaction by setting the corresponding inhibit input high, the note is returned immediately.

The sequence of operations is as follows:

- Pin 10 is held low awaiting note insertion
- Note inserted. Validator issues a 100 ms pulse on the appropriate channel
- The host machine initiates the vend process
- The host machine sets pin 10 high to indicate that it wants the note. If this is not done within 30 seconds the Validator will return the note
- The Validator issues a 100 ms pulse on the appropriate channel after pin 10 going high to indicate final acceptance of the note. If the signal has not been received within 30 seconds it indicates the customer has forcibly retrieved the note and the vend will be aborted
- The vend process is completed
- The host machine sets pin 10 low ready for the next vend operation

The host machine can force the return of the note to the customer by setting the inhibit line high at any time before the end of the 30 second time-out. For channels above 4 setting all inhibits high will cause a note reject.

In the event of a note being forcibly removed from the mouth of the NV9 USB during the 30 second interval, the NV9 USB will go out of service for 45 seconds.

5.6 SSP Escrow Function

To hold a note in the escrow position when using SSP, the POLL command should be replaced with the HOLD (0x18) command after NOTE READ > 0 for as long as the note is to be held in escrow.

A POLL (0x07) command will then accept the note; the REJECT (0x08) command will return the note to the customer

5.7 Credit Hold Function

This function is only available if the validator is set to operate in Pulse mode.

If the credit hold function is enabled (either by configuration card or BNV Currency Manager Program), the validator will take the note as normal but then wait until the escrow line is toggled low/high. It will then give out the number of pulses per note denomination as set when programmed. After the pulses have been generated, the validator will then wait for another low/high toggle until the full value of credit pulses are given.

As an example, with a setting of 4 pulses per banknote, a 5 euro note will give 4 pulses, 5 times. A typical use of this option would be for a pool table with a game price of €1. You could insert a €5 note and press a button that toggles the escrow line and releases the pool balls; this would then allow you to play the first game. The validator holds onto the remaining credits until the game has finished and the button is pressed again allowing the next game to begin, this continues until all the credits have been used.

The busy line remains low throughout the whole process and the validator remains inhibited until all pulses are given.

5.8 Connection Options

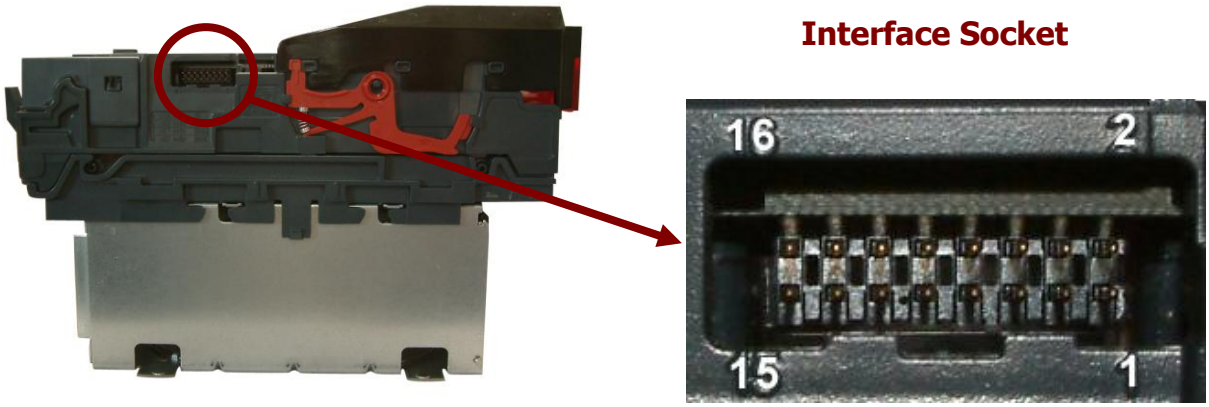
The NV9 USB Validator has a single connector that is used to allow interfacing and programming.

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 located on the side of the validator head. This connector is used to interface the NV9 USB 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 NV9 USB, a USB cable 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, ccTalk and SIO modes. When using the USB connection, power must be supplied to the NV9 USB using the CN00392 cable - further details of the cable needed to interface and program the NV9 USB validator can be found in Section 4 of this manual set (subsection 4.9).