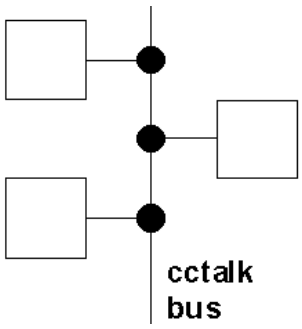


Lumina Serial Protocol

-

Issue 6.1

Revision History

Issue	Date	Comments
4.0	04-06-03	'core-4' production release
	05-08-03	Addition of escrow timeout
6.0	16-05-05	Additional timing information for 'Perform self-check'
6.1	18-05-05	TSP number added and released.

Warning : This manual covers core-4 Lumina to core-6 Lumina only.

To find out which core release you have send the cctalk command 'Request software revision' and look at character positions 9 and 10.

For example,
57x2035.03 is core 3.
57x2035.04.01 is core 4.
57x2035.06.13 is core 6.

Contents

1. Documentation

In order to write a host machine interface to the Lumina bill validator, the following additional documents will need to be read.

'cctalk Serial Communication Protocol - Generic Specification - Issue 4.2'

This is available from the cctalk web site at…
www.cctalk.org

'cctalk Expansion for Bill Validators - Issue 2.2'

This document is available from the Technical Support Department at Money Controls and contains cctalk extensions for interfacing to bill validators.

'cctalk Serial Protocol - Encryption Standard - Version 1.1'

This document is available from the Technical Support Department at Money Controls and contains details of the encryption algorithm used on cctalk bill validators. For security reasons this document cannot be emailed to you and will only be sent after certain checks have been conducted.

2. Introduction

The Lumina bill validator can operate entirely in serial mode with all conventional parallel control and status functions implemented in the cctalk serial protocol. This protocol is now a global standard in the money transaction industry.

This document contains details of operation in serial mode only. It is intended to be read by a software engineer wishing to interface to a Lumina from a PC or embedded system.

3. Encryption and Security

To meet the rigorous requirements for fraud prevention on bill validators, Lumina **only operates in serial mode with the encryption layer and CRC checksum enabled.** These are described fully in the documents listed previously. This affects all cctalk headers on Lumina - there is no provision in the protocol for a mixture of insecure and secure commands.

So cctalk software written to communicate with the SR range of coin acceptors or earlier, and the Mk1 & Mk2 serial compact hoppers, will not work without an additional software layer in the protocol. It is possible to mix unencrypted and encrypted devices on the cctalk bus due to the nature of the packet structure.

No communication with Lumina is possible without knowledge of the 6 digit security key.

The Lumina supplied should have small grey-silver label on top with a 6 digit security code which is the one needed to communicate with the product. If it does not then you can assume the product was initialised with a test key value of '123456'. Money Controls can also trace the security code given the product serial number.

For applications that don't require a great deal of security, the key value can be left unchanged. Otherwise, the key should be changed every few seconds by the host machine software. For even more security, the new key value can be stored perhaps every 15 minutes inside Lumina so that next time the machine powers-up, Lumina starts off with a new and seemingly random key value. To protect against power fails, the host machine must store the new key value in its own non-volatile memory before attempting to store it on Lumina.

4. Protocol Descriptor

The Lumina serial protocol conforms to cctalk b96.p0.v12.a5.d0.c5.m0.x16.e1.i2.r4

In other words…

- 9600 baud (1 start bit, 1 stop bit, no parity)
- Open-collector interface
- Nominal supply voltage +12V
- Serial data pull-up voltage +5V
- Supply sink
- Connector type 5 (10-way pin header)
- Slave device only
- CRC CCITT checksum
- Encryption type 1
- Minor release 2
- Major release 4

5. Device Address

All Lumina's leave the factory set to address **40**.

The address is stored in EEPROM and can be subsequently changed with serial commands. Unless you have an application requiring more than one bill validator on the serial bus, it is strongly recommended you leave the address alone. The default addresses for coin acceptors and hoppers have been made different and will not clash with the bill validator.

If the address has been changed to an unknown value then you will either have to search through the entire address space (2 to 255) with the 'Simple poll' command until an ACK is returned or send the 'Address poll' command with the broadcast address.

6. Electrical Connections

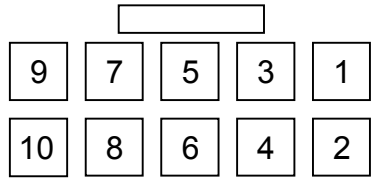
There are 2 types of cctalk serial connector being used on Lumina depending whether it is a ‘new’ or ‘old’ PCB. The older PCB’s are supplied with a cable converter which changes the connector into the new type.

The new PCB connector is a UK BACTA-approved, 10-way, dual-row, pin header on a 0.1inch pitch.

6.1 New PCB

(1) **/DATA**
(2) <reserved>
(3) <reserved>
(4) <reserved>
(5) <reserved>
(6) <reserved>
(7) **+Vs**
(8) **0V**
(9) <reserved>
(10) <reserved>

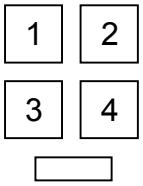
Note that no connections are made to the /SERIAL MODE function on pin 9 as serial mode is selected on the DIP switches.



View of PCB connector from front

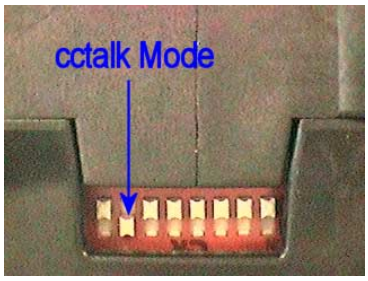
6.2 Old PCB

(1) **+Vs**
(2) <reserved>
(3) **0V**
(4) **/DATA**



View of socket from front

7. DIP Switches



To operate Lumina in cctalk mode, ensure that the device powers up with DIP switch 7 in the ON position. All other switches MUST BE OFF.

8. Inhibits

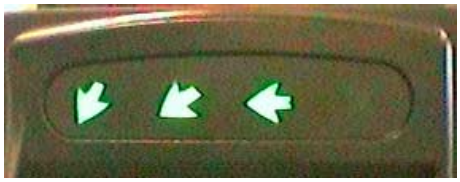
At power-up or after a reset, all bills on Lumina are inhibited and a red cross is shown on the lamp panel.

The motor transport system does not operate until at least one bill is enabled. This is achieved by lifting the master inhibit as well as a number of the individual bill inhibits.

The cctalk commands required to do this are…

Header 231 - Modify inhibit status
Header 228 - Modify master inhibit status

When the Lumina is ready to take a bill, the lamp panel shows 3 right to left sequencing green arrows.



It is only possible to have green arrows if there are programmed bills within Lumina.

9. Credit Polling

Bill credits are obtained by polling the bill validator at regular intervals using the 'Read buffered bill events' command. Up to 5 credits (or events) are stacked in the return buffer.

The return data is 11 bytes in length.

[event counter]
[result 1A] [result 1B]
[result 2A] [result 2B]
[result 3A] [result 3B]
[result 4A] [result 4B]
[result 5A] [result 5B]

Note that as per the generic specification, the event counter wraps from 255 to 1 during normal operation. A value of zero indicates a power-down or reset has occurred with the possible loss of some credits.

9.1 Credit Poll Timeout

Lumina has a safety mechanism such that if the unit is not polled with cctalk header 159 for a specified time then it inhibits itself and cannot take any more bills. This prevents bill swallowing if a fault develops with the host machine.

The 'Master inhibit active' event is added to the event buffer when the inhibit condition is latched. The individual bill inhibits are left unchanged but the master inhibit flag is set. To resume normal operation a 'Modify master inhibit status' command needs to be sent.

After the Lumina powers up, the inhibits need to be lifted before bills can be accepted. However, the polling operation must resume within the 'credit poll timeout' period or the inhibits will need to be lifted again.

9.2 Typical Event Codes

These are some typical 2-byte event codes as reported by the 'Read buffered bill events' command on Lumina in escrow mode.

True Bill Insertion

[1] [1] - bill type 1 held in escrow
(Route bill command issued)
[1] [0] - bill type 1 sent to cashbox

Reject Bill Insertion

[0] [2] - invalid bill
[0] [1] - bill returned from escrow

Bill Inserted Incorrectly

[0] [3] - transport error

Fraud Attempt

[0] [9] - bill tamper
or
[0] [18] - string fraud

10. Escrow Operation

Lumina uses escrow mode by default in cctalk serial mode. This means that an inserted bill is not sent directly into the cashbox but is held in the 'escrow' position while the host machine is notified of the pending credit. The host machine can then make the decision to accept the bill or to reject it. This is done with header 154, 'Route bill'. If no routing decision is made within the 'escrow timeout' period then the bill is returned to the customer.

If escrow mode is not wanted, it can be switched off with header 153, 'Modify bill operating mode'. This change is maintained until power-down or reset.

11. Software Design Notes

1. It is essential that a retry mechanism is written into the driver software, especially when polling for events with header 159. Due to processor loading, when a bill is in transport, an occasional poll command is ignored. It is suggested there is an automatic retry after 50ms.
2. When testing the parallel vend lines using header 238, 'Test output lines', the Vend 1 line is 'stuck on'. This is a consequence of the CPLD hardware but does not affect serial mode operation.
3. Only a single bank of bills is supported so there is no support for headers 178 and 179.
4. There is no support currently for teaching bills using headers 201 and 202.
5. There is no support currently for changing bill security using headers 180 and 181.
6. Individual counting of bills using header 150 is not supported.
7. Individual counting of errors using header 149 is not supported.
8. There is no Lumina stacker.

12. Key Timing Parameters

Description	Time
Power-up : Initialisation time	1s
Power-up : Can send 'Perform self-check' command	10s
Power-up : Ready to accept bills time	10s
Software reset : Ready to accept bills time	10s
Recommended credit polling interval	200ms
Recommended dynamic key change interval	>1s
Credit poll timeout	5s
Escrow timeout	30s

Notes
During the power-up initialisation time Lumina clamps the cctalk data line low for up to 1s which prevents serial communication with any other peripheral on the bus. After initialisation, cctalk commands can be sent to Lumina but bills should not be enabled until the device is ready to accept. Otherwise, the master inhibit will be set and no bills will be accepted.

The recommended dynamic key change interval prevents excessive processor loading during bill transport and validation.

Note that the diagnostics command 'Perform self-check' should not be sent until 10s after a power-up or reset as the Lumina performs a calibration operation on the sensors which interferes with diagnostics. Any fault code returned during this initialisation period should be ignored.

The correct sequence of events is…
<power-up> <wait 10s> <self-check> <enable master inhibit> <enable individual bills> <poll for events>

It is also recommended that the 'Perform self-check' command is not sent during note transport for the same reason.

13. Typical Command Response Times

Examples are of cctalk response times are shown in the table below. They are meant to be indicative only.

Header	Description	Total TX + RX Packet Length	Typical Response Time
136	Store encryption code	10	95 ms
137	Switch encryption code	13	55 ms
157	Request bill id	18	34 ms
156	Request country scaling factor	15	24 ms
197	Calculate ROM checksum	14	22 ms
246	Request manufacturer id	13	21 ms
242	Request serial number	13	19 ms
226	Request insertion counter	13	19 ms
231	Modify inhibit status	12	18 ms
232	Perform self-check	11	17 ms
228	Modify master inhibit status	11	16 ms
254	Simple poll	10	14 ms
001	Reset device	10	14 ms
159	Read buffered bill events	21	3 ms

The typical response time is measured from the final stop bit of the transmitted message to the start bit of the reply. This is the time taken for Lumina to decrypt the host message, check the CRC, perform the action requested (which could in theory take anything from milliseconds to seconds), prepare the return data, calculate the CRC and encrypt the reply.

14. Debugging : No reply to any command ?

There are various reasons why there may be no reply to a cctalk serial command. Try checking the following items.

14.1 Encryption & Security Key

All Lumina commands are protected by the encryption layer in the protocol. Unless the exact security key is known, no communication is possible. For prototype units, try the test key of '123456'.

Try sending a 'Simple poll' command to Lumina. This is a short cctalk command with no transmitted data and no received data (ACK only).

The complete message prior to encryption will be…
[40] [0] [182] [254] [33]

After encryption the message depends on the security key. Assuming the encryption key is '123456' the message will be…
[40] [0] [127] [192] [137]

The reply will be…
[1] [0] [88] [56] [230]

After decryption this is…
[1] [0] [48] [0] [55] = ACK

Do the additional examples in the encryption manual validate successfully ?

14.2 Checksum

Lumina uses a 16-bit CRC checksum. Unless this is calculated correctly there will be no reply. Check your code with the examples in the BNV specification.

14.3 Power

Kind of obvious but is the Lumina connected to a regulated +12V supply with sufficient current ? Note also that cctalk commands cannot be sent immediately after power-up.

14.4 Address

The default Lumina address is 40 but this can be changed by writing to the configuration memory. Try using the broadcast address (= 0) with no other devices connected to the bus. Is there a reply ? If so, try using header 253, 'Address poll' to find out the address.

15. Command List

The following table shows all the cctalk commands currently available for customer use on Lumina. More details can be found in the generic specification.

Any [data bytes] are shown in decimal.

MDCES = Multi-Drop Command Extension Set. These commands are only used when peripheral addresses are unknown.

Header numbers are shown in descending order.

Header	Function	Returned Data / Comments
254	Simple poll	ACK returned.
253	Address poll	MDCES support
252	Address clash	MDCES support
251	Address change	MDCES support
250	Address random	MDCES support
249	Request polling priority	[1] [200] = 200 ms
247	Request variable set	[No. of bills supported] = 16 [No. of banks supported] = 1 [No. of channels supported] = 64
246	Request manufacturer id	'MCI' for Money Controls International
245	Request equipment category id	'Bill Validator'
244	Request product code	'Lumina'
242	Request serial number	Range 0 to 16,777,215
241	Request software revision	e.g. '57x2035.06.13' core 06 software revision 13
238	Test output lines	Send [Validator Output Mask] B0 - Vend 1 B1 - Vend 2 B2 - Vend 3 B3 - Vend 4 B4 - Alarm A single byte parameter allows the parallel output lines to be tested. The outputs are pulsed for 500ms. The ACK is returned after the output pulse has finished.

237	Read input lines	[DIL Switch] Each bit represents an individual switch B0 - Switch 1 … B7 - Switch 8 Polarity: 0 - off (up) 1 - on (down) [XXX | Escrow Accept | Inhibits] B0 - Inhibit 1 (0 = inhibit active) B1 - Inhibit 2 B2 - Inhibit 3 B3 - Inhibit 4 B4 - Escrow Accept (0 = accept) This command can be used to verify all the input lines to the processor are working correctly.
236	Read opto states	[opto states] B0 - Bill Sensing Left Front B1 - Bill Sensing Right Front B2 - Bill Sensing Left Rear B3 - Bill Sensing Right Rear Polarity: 0 - opto clear 1 - opto blocked
232	Perform self-check	Supported. Refer to table 3 in the generic specification for possible fault codes.
231	Modify inhibit status	Send [inhibit 1] [inhibit 2] Support for 16 bills where… 0 = bill inhibited, 1 = bill enabled Inhibits are stored in RAM and are lost at power-down or reset. The power-up state is ‘all bills inhibited’.
230	Request inhibit status	[inhibit 1] [inhibit 2] The inhibit status of 16 bill positions is returned
228	Modify master inhibit status	0 = device inhibited, 1 = device enabled The master inhibit flag is stored in RAM and is cleared at power-down or reset. The power-up state is ‘master inhibited’.
227	Request master inhibit status	Supported

226	Request insertion counter	This counter records the total number of inserted bills. The counter is stored in EEPROM and its value is retained after a power-down or reset. The counter value is stored every 10 insertions and so should be used as an approximate guide to performance rather than for auditing purposes.
225	Request accept counter	This counter records the total number of accepted bills. The counter is stored in EEPROM and its value is retained after a power-down or reset. The counter value is stored every 10 accepts and so should be used as an approximate guide to performance rather than for auditing purposes.
216	Request data storage availability	[2] [2] [8] [2] [8] This indicates a small amount of EEPROM storage is available for customer use… 2 blocks of 8 bytes for reading 2 blocks of 8 bytes for writing
215	Read data block	Block number = 0 or 1
214	Write data block	Block number = 0 or 1
213	Request option flags	[Option Flags] Stacker supported = No Escrow supported = Yes Individual accept counters = No Individual error counters = No Non-volatile counters = Yes Bill teach supported = No Bill security supported = No Remote bill programming = Yes
197	Calculate ROM checksum	An internal ROM checksum is returned to the host machine. [XX] [XX] [0] [0]
196	Request creation date	Supported
195	Request last modification date	Supported
192	Request build code	8 character ASCII string e.g. ‘Standard’
170	Request base year	‘2001’
169	Request address mode	[132] Address is stored in EEPROM and may be changed serially (non-volatile).
159	Read buffered bill events	[event counter] [result 1A] [result 1B] [result 2A] [result 2B] [result 3A] [result 3B] [result 4A] [result 4B] [result 5A] [result 5B] 5 event buffer
157	Request bill id	Send [bill type] = 1 to 16 If returned string is ‘…….’ (7 x ASCII 46) then no bill is programmed in that position.
156	Request country scaling factor	Supported

155	Request bill position	Supported
154	Route bill	Supported
153	Modify bill operating mode	Stacker is not supported and cannot be enabled Escrow mode can be switched on and off. The default / power-up state is escrow on.
152	Request bill operating mode	Supported
151	Test lamps	Send [lamp no.] 1 - Red cross 2 - Green arrow right 3 - Green arrow middle 4 - Green arrow left If any lamp is set to automatic then the normal display sequence resumes.
148	Read opto voltages	Supported 22 bytes of opto voltage data are returned for diagnostic purposes. No further details are given here.
146	Operate bi-directional motors	Send [motor bit mask] Send [direction flags] Send [speed] Only B0 is supported for the bill transport motor. Forward and reverse directions are supported. Only 'normal' speed is supported - there is no user PWM control.
145	Request currency revision	Supported. Rather than a global currency revision, the revision of each programmed bill can be displayed. e.g. '010101. . . . . ' Each of the 16 bill positions has a 2 digit revision code 00 to 99. The first 2 digits are for position 1, the last 2 for position 16. 32 characters in total. An unprogrammed position is displayed as '. ' (ASCII 46, 32). A 2 character country code must be provided as part of the tx data.
144	Upload bill tables	Supported
143	Begin bill table upgrade	Supported
142	Finish bill table upgrade	Supported
141	Request firmware upgrade capability	[1] = firmware in FLASH / EEPROM with upgrade capability The firmware in Lumina is held in FLASH memory and may be field-updated.
140	Upload firmware	Supported
139	Begin firmware upgrade	Supported
138	Finish firmware upgrade	Supported

137	Switch encryption code	Supported
136	Store encryption code	Supported
4	Request comms revision	[1] [4] [2] cctalk specification = 4.2 minor revision = 1
3	Clear comms status variables	Supported
2	Request comms status variables	Supported
1	Reset device	Supported This command performs a 'software reset'. All bills are inhibited afterwards.

16. Event Code Table

This table is taken from the cctalk generic specification.

Shaded = Implemented on Lumina

Result A	Result B	Event	Type
0	0	Master inhibit active	Status
0	1	Bill returned from escrow	Status
0	2	Invalid bill (due to validation fail)	Reject
0	3	Invalid bill (due to transport problem)	Reject
0	4	Inhibited bill (on serial)	Status
0	5	Inhibited bill (on DIP switches)	Status
0	6	Bill jammed in transport (unsafe mode)	Fatal Error
0	7	Bill jammed in stacker	Fatal Error
0	8	Bill pulled backwards	Fraud Attempt
0	9	Bill tamper	Fraud Attempt
0	10	Stacker OK	Status
0	11	Stacker removed	Status
0	12	Stacker inserted	Status
0	13	Stacker faulty	Fatal Error
0	14	Stacker full	Status
0	15	Stacker jammed	Fatal Error
0	16	Bill jammed in transport (safe mode)	Fatal Error
0	17	Opto fraud detected	Fraud Attempt
0	18	String fraud detected	Fraud Attempt

17. Fault Code Table

This table is taken from the cctalk generic specification.

Shaded = Implemented on Lumina

Code	Fault	Optional Extra Info
0	OK (no fault detected)	-
1	EEPROM checksum corrupted	-
30	ROM checksum mismatch	-
36	Fault on bill validation sensor	Identify which sensor
37	Fault on bill transport motor	-
38	Fault on stacker	-
39	Bill jammed	-
40	RAM test fail	-
41	Fault on string sensor	-
255	Unspecified fault code	-

18. Remote Bill Programming

18.1 Introduction

It is possible to update or completely replace the programmed bills in Lumina with a different currency remotely. This means through the cctalk serial interface and without the manual insertion of any bills.

4 cctalk commands are associated with remote bill programming.

- Header 145, Request currency revision
- Header 144, Upload bill tables
- Header 143, Begin bill table upgrade
- Header 142, Finish bill table upgrade

18.2 Is Remote Bill Programming Supported ?

To check whether the Lumina you have supports remote bill programming…

1. Send cctalk command 'Request software revision'

The returned ASCII string will be something like 57x2035.06.13. The digits 06 are the core software release version. Remote bill programming is only supported on core 4 and higher.

2. Send cctalk command 'Request option flags'

Bit 7 of the returned status byte indicates (if set) whether remote bill programming is supported.

18.3 Software Engineers : Please Be Aware !

Due to the software overheads associated with programming flash memory, it is not possible to enable the cctalk encryption layer during bill programming. CRC checksums are still used however so the host driver software just needs a way of enabling / disabling encryption during programming.

So **disable encryption during bill programming**.

Use address 40 during bill programming. Do not use the broadcast address.

Always reset Lumina after programming or erasing bills with cctalk header 1.

Do not mix entire currency files with individual bills. In other words, do not program an entire currency file then change a few individual bill entries. This could result in a poor to no acceptance of true bills and a poor rejection of fraud bills. Use one method or the other. When changing from an entire currency file to one programmed individually, make sure any unprogrammed bill positions are **always erased**.

18.4 Programming Procedure

18.4.1. Entire Currency

A typical filename for a currency is **GB01.tab**. Size 50.5K. It contains the patterns for all bills within that currency. The '01' is a configuration number as there may be some options on which 16 bills in that currency have been pre-selected for programming.

The data inside each file is in binary format. The entire file must transferred to Lumina, 128 bytes at a time. No modifications to the data must be made.

Step 1. Send cctalk command 'Begin bill table upgrade'
TX data : < none >
RX data : ACK
Wait up to 3s for reply

Step 2. Switch encryption off

Step 3. Send cctalk command 'Upload bill tables' header

Format of TX data is [block] [line] [data 1] [data 2]… [data N] where N <= 128

TX data : [0] [0] [1] [0] [0]
RX data : ACK
Wait up to 2s for reply

Block 0, line 0 begins the upload sequence.

Step 4. Send cctalk command 'Upload bill tables' data

```
For block = 1 to last_block
        For line = 0 to 255
                TX data : [ block ] [ line ] [ data 1 ]… [ data 128 ]
                RX data : ACK
                Wait up to 2s for reply
        Next
Next
```

Loop until all the file data is sent. The last line may contain less than 128 bytes of data.

Step 5. Send cctalk command 'Finish bill table upgrade'

TX data : < none >
RX data : ACK
Wait up to 1s for reply

Step 6. Switch encryption on

Step 7. Wait 3s and send cctalk command 'Reset device'

Wait 3s
<reset>

After10s Lumina can be enabled and bills accepted.

18.4.2. Individual Bills

A typical filename for a bill is **EU0020A1.tab**. Size 3.3K. It contains the patterns for just one bill in all 4 orientations.

The data inside each file is in binary format. The entire file must transferred to Lumina, 128 bytes at a time. No modifications to the data must be made.

When programming individual bills, the bill may be programmed into any of the 16 'free slots' on Lumina. If there is already a bill at that position then it will be overwritten. There is a way of erasing bills - see next section.

Step 1. Send cctalk command 'Begin bill table upgrade'
TX data : < none >
RX data : ACK
Wait up to 3s for reply

Step 2. Switch encryption off

Step 3. Send cctalk command 'Upload bill tables' header

Format of TX data is [block] [line] [data 1] [data 2]… [data N] where N <= 128

TX data : [0] [0] [0] [bill position] [0]
RX data : ACK
Wait up to 2s for reply

[bill position] = 1 to 16

Block 0, line 0 begins the upload sequence.

Step 4. Send cctalk command 'Upload bill tables' data

For block = 1 to last_block
    For line = 0 to 255
        TX data : [block] [line] [data 1]… [data 128]
        RX data : ACK
        Wait up to 2s for reply
    Next
Next

Loop until all the file data is sent. The last line may contain less than 128 bytes of data.

Step 5. Send cctalk command 'Finish bill table upgrade'

TX data : < none >
RX data : ACK
Wait up to 1s for reply

Step 6. Switch encryption on

Step 7. Wait 3s and loop back for other bills

Wait 3s
Go to Step1 until all bills have been programmed.

Step 8. Send cctalk command 'Reset device'

<reset>

After10s Lumina can be enabled and bills accepted.

18.4.3. Erasing Bills

The method used to erase a bill position on Lumina is to send a blank file called 'Erase.tab'. The programming sequence is exactly the same as for individual bills. An erase file will be provided which can work with any downloadable bill set you use.

There is a mechanism in the 'Upload bill tables' header to erase a bill directly but this has not been implemented in the core 6 release.

TX data : [0] [0] [2] [bill position] [0]
RX data : ACK

18.4.4. Individual Bill Filenames

By example…

EU0020A1.tab

The filename uses a 8.3 convention.

EU	2 character country code as listed in ISO 3166-1-A2
0020	4 digit monetary value in decimal
A	Mint issue letter
1	Sub-issue code
tab	Indicates bill table file

The monetary value is used in conjunction with the country scaling factor (see cctalk header 156, 'Request country scaling factor') to determine the absolute value when using coins and bills in the same machine.

On bills the mint issue letter changes when the 'picture' on either side of the bill changes. This could be for when new bills replace older ones or when different mints or banks issue the same bill (e.g. Scottish bills).

An Excel spreadsheet of the latest bill names can be found on www.cctalk.org

A sub-issue code is also included which reflects 'invisible' changes i.e. those not obvious by looking at the bill. These may be changes in magnetic inks or special dyes, or even fraud information which is included alongside the true bill table. When a bill is issued with 2 or more sub-issue codes, they should ALL be programmed sequentially into the bill validator to maximise fraud performance.

e.g. if you are supplied with GB0005A1, GB0005A2, GB0010A1 then program as follows…
GB0005A1 ➔ Position 1
GB0005A2 ➔ Position 2
GB0010A1 ➔ Position 3

You can see that the GB5A bill actually occupies 2 positions out of 16 leaving you less room for programming additional bills. For this reason the GB currency is often issued as an entire currency file which has a more efficient packing mechanism.

18.4.5. DOS Stub

The first 13 characters of both the entire currency file and the individual bill file are in ASCII and may be viewed under DOS by typing the filename to the screen (or in Windows by opening them with Notepad).

e.g. Type EU0020A1.tab

B LUMINA01 01

= B(ill Validator) Lumina 01 bills revision 01

An entire currency file may contain B LUMINA16 01 for 16 bills.

18.5 Typical Reprogramming Times

These are typical times only using the same delays as the examples above.

Entire Currency	1 min 18 secs
16 x Individual Bills	2 mins 25 secs
Individual Bills	9 secs per bill

18.6 Currency Revision Codes

The currency revision is returned with cctalk header 145, 'Request currency revision'.

The currency code of interest should be transmitted in the data packet.

TX : ['G'] ['B']
RX : [Char 1] [Char 2]… [Char 32]

Each of the 16 bill positions has a 2 digit revision code 00 to 99. The first 2 digits are for position 1, the last 2 for position 16. 32 characters in total. An unprogrammed position is displayed as '. ' (ASCII 46, 32).

Example

```
010101. . . . . . . . . . . . . . .
```

Bill	Revision
1	01
2	01
3	01
4	Blank
5	Blank
6	Blank
7	Blank
8	Blank
9	Blank
10	Blank
11	Blank
12	Blank
13	Blank
14	Blank
15	Blank
16	Blank

19. Firmware Upgrading

19.1 Introduction

It is possible to upgrade the firmware flash memory on Lumina through the cctalk serial interface.

4 cctalk commands are associated with firmware upgrading.

- Header 141, Request firmware upgrade capability
- Header 140, Upload firmware
- Header 139, Begin firmware upgrade
- Header 138, Finish firmware upgrade

19.2 Is Firmware Upgrading Supported ?

To check whether the Lumina you have supports firmware upgrading…

1. Send cctalk command 'Request software revision'

The returned ASCII string will be something like 57x2035.06.13. The digits 06 are the core software release version. Firmware upgrading is only supported on core 4 and higher.

2. Send cctalk command 'Request firmware upgrade capability'

The returned data byte should be [1] indicating firmware upgrading is possible.

19.3 Software Engineers : Please Be Aware !

Due to the software overheads associated with programming flash memory, it is not possible to enable the cctalk encryption layer during firmware upgrading. CRC checksums are still used however so the host driver software just needs a way of enabling / disabling encryption during upgrading.

So **disable encryption during firmware upgrading**.

Use address 40 during firmware upgrading. Do not use the broadcast address.

Always reset Lumina after upgrading with cctalk header 1.

Upgrading firmware erases all programmed bills. Bills must be reprogrammed afterwards using the methods described previously.

19.4 Programming Procedure

A typical filename for a firmware release is **57x2035.06.13.bin**. Size 128K.

The data inside each file is in binary format. The entire file must transferred to Lumina, 128 bytes at a time. No modifications to the data must be made.

Step 1. Send cctalk command 'Begin firmware upgrade'
TX data : < none >
RX data : ACK
Wait up to 3s for reply

Step 2. Switch encryption off

Step 3. Send cctalk command 'Upload firmware'

Format of TX data is [block] [line] [data 1] [data 2]… [data N] where N <= 128

For block = 0 to last_block
    For line = 0 to 255
        TX data : [block] [line] [data 1]… [data 128]
        RX data : ACK
        Wait up to 2s for reply
    Next
Next

Loop until all the file data is sent. The last line may contain less than 128 bytes of data.

Step 4. Send cctalk command 'Finish firmware upgrade'

TX data : < none >
RX data : ACK
Wait up to 1s for reply

Step 5. Switch encryption on

Step 6. Wait 3s and send cctalk command 'Reset device'

Wait 3s
<reset>

After10s Lumina can be enabled and bills accepted.

19.5 Typical Reprogramming Times

These are typical times only using the same delays as the examples above.

New Firmware	3 mins 2 secs

20. Appendix A : Key Recovery

In the unfortunate event that you change the security key with cctalk header 137, 'Switch encryption code', save the security key with cctalk header 136, 'Store encryption code' and then lose the key, there is a way of recovering the factory preset value - as shown on the encryption label. Until you do this it is not possible to communicate with the Lumina serially.

Switch power off the device and turn DIL switch 6 on. Power up the device, wait a few seconds and then power down. Turn DIL switch 6 off and then power up the device as normal. The security key will have reset.

A good algorithm for verifying operation after a key change is to try the new key value first. If that doesn't work try the old key value. If that doesn't work try the original key value (when the Lumina was first connected to the host machine). If that doesn't work you will need to remove the Lumina from the machine and recover the key as described above.

21. Appendix B : Example Serial Capture Log

An example serial capture log is shown below for a host machine communicating with a Lumina bill validator. It includes the insertion of a single note in escrow mode.

The data values in magenta have been decrypted.

The command sequence is as follows…

- Request variable set - find out how many bills there are in the device (= 16)
- Request bill id - loop through positions 1 to 16 and find out what bills are programmed
- Request country scaling factor - calculate bill values
- Modify inhibit status - enable all bills for acceptance
- Modify master inhibit status - enable the bill validator (lamps change to green)
- Read buffered bill events - wait for notes to be inserted
- Switch encryption code - change the security key at regular intervals to confuse the enemy
- Route bill - route true bills into cashbox

```
(1) Encrypted
  +1004ms - 040 000 159 247 176 - OK
From Machine : Request variable set
    +23ms - 001 003 228 000 016 001 064 147 - OK
From Device  : Return message header

(2) Encrypted
    +55ms - 040 001 043 157 001 251 - OK
From Machine : Request bill id
    +33ms - 001 007 049 000 071 066 048 048 048 053 065 188 - OK
From Device  : Return message header
Bill id = GB0005A

(3) Encrypted
    +43ms - 040 001 072 157 002 203 - OK
From Machine : Request bill id
    +33ms - 001 007 244 000 071 066 048 048 049 048 065 116 - OK
From Device  : Return message header
Bill id = GB0010A

(4) Encrypted
    +43ms - 040 001 105 157 003 219 - OK
From Machine : Request bill id
    +33ms - 001 007 164 000 071 066 048 048 050 048 065 045 - OK
From Device  : Return message header
Bill id = GB0020A

(5) Encrypted
    +43ms - 040 001 142 157 004 171 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(6) Encrypted
```

```
    +43ms - 040 001 175 157 005 187 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(7) Encrypted
    +43ms - 040 001 204 157 006 139 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(8) Encrypted
    +43ms - 040 001 237 157 007 155 - OK
From Machine : Request bill id
    +32ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(9) Encrypted
    +44ms - 040 001 002 157 008 106 - OK
From Machine : Request bill id
    +32ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(10) Encrypted
    +44ms - 040 001 035 157 009 122 - OK
From Machine : Request bill id
    +32ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(11) Encrypted
    +43ms - 040 001 064 157 010 074 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(12) Encrypted
    +43ms - 040 001 097 157 011 090 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(13) Encrypted
    +44ms - 040 001 134 157 012 042 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(14) Encrypted
    +43ms - 040 001 167 157 013 058 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(15) Encrypted
    +42ms - 040 001 196 157 014 010 - OK
```

```
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(16) Encrypted
    +43ms - 040 001 229 157 015 026 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(17) Encrypted
    +43ms - 040 001 059 157 016 249 - OK
From Machine : Request bill id
    +33ms - 001 007 177 000 046 046 046 046 046 046 046 024 - OK
From Device  : Return message header
Bill id = .......

(18) Encrypted
    +70ms - 040 002 116 156 071 066 022 - OK
From Machine : Request country scaling factor
    +27ms - 001 003 155 000 100 000 002 204 - OK
From Device  : Return message header

(19) Encrypted
     +0ms - 040 002 095 231 255 255 223 - OK
From Machine : Modify inhibit status
Inhibits MSB 11111111-11111111 LSB
    +22ms - 001 000 048 000 055 - OK
From Device  : ACK

(20) Encrypted
    +60ms - 040 001 234 228 001 073 - OK
From Machine : Modify master inhibit status
Master inhibit OFF
    +20ms - 001 000 048 000 055 - OK
From Device  : ACK

(21) Encrypted
   +455ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(22) Encrypted
   +457ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(23) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(24) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
```

```
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(25) Encrypted
   +122ms - 040 003 110 137 087 037 115 148 - OK
From Machine : Switch encryption code
    +28ms - 001 000 048 000 055 - OK
From Device  : ACK
Security key changed to 755237

(26) Encrypted
   +220ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
    +38ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(27) Encrypted
   +306ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(28) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(29) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(30) Encrypted
   +122ms - 040 003 235 137 112 120 064 233 - OK
From Machine : Switch encryption code
    +29ms - 001 000 048 000 055 - OK
From Device  : ACK
Security key changed to 078704

(31) Encrypted
   +220ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
    +41ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(32) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(33) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
```

```
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(34) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(35) Encrypted
   +123ms - 040 003 186 137 057 117 080 014 - OK
From Machine : Switch encryption code
    +30ms - 001 000 048 000 055 - OK
From Device  : ACK
Security key changed to 935705

(36) Encrypted
   +232ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
    +41ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(37) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 158 000 000 000 000 000 000 000 000 000 000 000 000 191
- OK
From Device  : Return message header

(38) Encrypted
  +1209ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 059 000 001 003 001 000 000 000 000 000 000 000 000 061
- OK
From Device  : Return message header
Bill type 3 held in escrow

(39) Encrypted
    +70ms - 040 001 188 154 001 098 - OK
From Machine : Route bill
    +22ms - 001 000 048 000 055 - OK
From Device  : ACK

(40) Encrypted
   +307ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 246 000 002 003 000 003 001 000 000 000 000 000 000 222
- OK
From Device  : Return message header
Credit = 3

(41) Encrypted
   +329ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
     +0ms - 001 011 246 000 002 003 000 003 001 000 000 000 000 000 000 222
- OK
From Device  : Return message header

(42) Encrypted
   +316ms - 040 000 049 159 093 - OK
From Machine : Read buffered bill events
```

```
     +0ms - 001 011 246 000 002 003 000 003 001 000 000 000 000 000 000 222
- OK
From Device  : Return message header

(43) Encrypted
   +123ms - 040 003 121 137 136 149 041 056 - OK
From Machine : Switch encryption code
    +30ms - 001 000 048 000 055 - OK
From Device  : ACK
Security key changed to 885992
```

22. Appendix C : Software Issues

No issues have been reported.