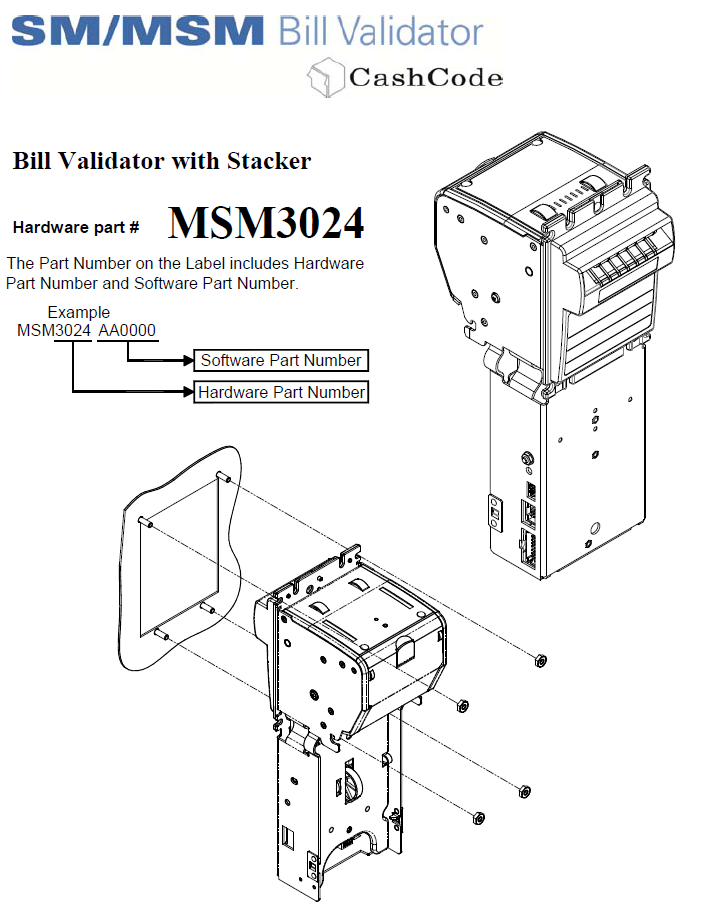
**CashCode MSM-3024 Bill Validator**

Getting Started Guide



[1 Basic Information 1](#_Toc62053578)

[1.1 Introduction 1](#_Toc62053579)

[1.2 Power requirements 1](#_Toc62053580)

[1.3 Communication 1](#_Toc62053581)

[2 Initial setup 4](#_Toc62053582)

[2.1 Powering up and establishing connection 4](#_Toc62053583)

[2.2 Communicating with the device 5](#_Toc62053584)

[2.2.1 Important Notes 5](#_Toc62053585)

[2.2.2 Communication format 5](#_Toc62053586)

[2.2.3 Commands 7](#_Toc62053587)

[2.2.4 POLL Command 8](#_Toc62053588)

[3 Communication Examples 10](#_Toc62053589)

[3.1 Example Messages 10](#_Toc62053590)

[3.1.1 POLL 10](#_Toc62053591)

[3.1.2 ACK 10](#_Toc62053592)

[3.1.3 STATUS 10](#_Toc62053593)

[3.1.4 Disable Bill Accepting 10](#_Toc62053594)

[3.2 Setup after powering on 11](#_Toc62053595)

[3.3 Bill Accepting Sequence 13](#_Toc62053596)

# Basic Information

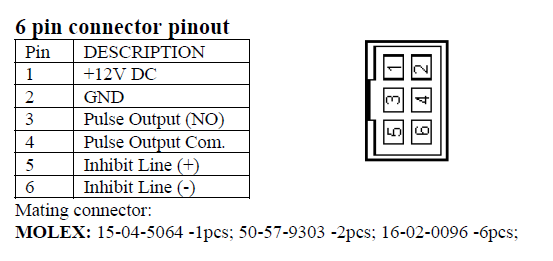
## Introduction

MSM3024 is a cash validator with integrated firmware with bill identification, verification and stacking capabilities. The unit is compatible with multiple types of stackers (a.k.a. cassettes) to store the bills received and validated by the validator. The unit has a front, bi-colour set of LEDs to indicate the user of the operational status of the unit.

* This unit (firmware LK1113) contains a firmware that can detect and identify Sri Lankan currency from Rs.10 to Rs.1000 notes. The types of notes accepted by the device can be selected through a certain set of commands, and can be overridden by selecting the DIP switches found beneath the stacker dock, on the motherboard.

## Power requirements

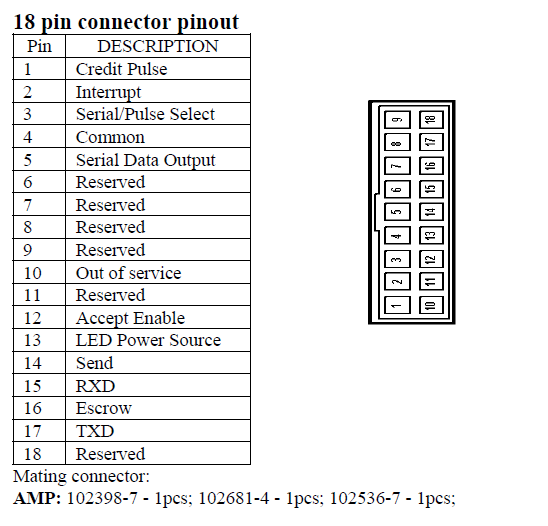
* This unit operates at 12VDC (11-14V accepted), and has a peak current draw of 2A when the unit is performing certain operations. Idling current is 800mA. The power should be regulated and a switched-mode power supply is preferred.



* The unit only requires +12VDC (Yellow) and GND(Black) wires to be powered on. 3, 4, 5, 6 pins can be left floating.

## Communication

* MSM-3024 communicated with external peripherals through an RS-232 serial connection interface at a user-selectable baud rate of 9600bps or 19200bps. This is hardware selected and can be changed through the DIP switches. The device requires all the communication done according to the CCNet protocol, of which the command reference can be obtained through [this](https://kiosksoft.ru/device_files/download/87) link.



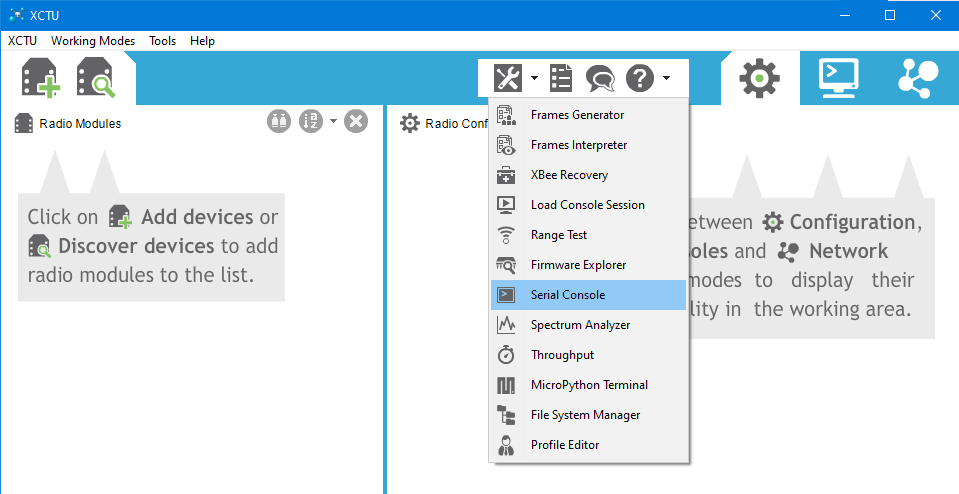
* The supplied cable only connects to +5V, TXD, RXD and GND pins of the 18-pin connector, and the other end of the cable is connected to a DB-9 female connecter which plugs into the USB-RS232 converter.
* Firmware updates of the device can be carried out through MDB interface.
* Computers/controller devices/master devices that do not have a built-in RS232 serial interface must use an intermediate USB-to-RS232 serial converter which has a DB-9 interface.

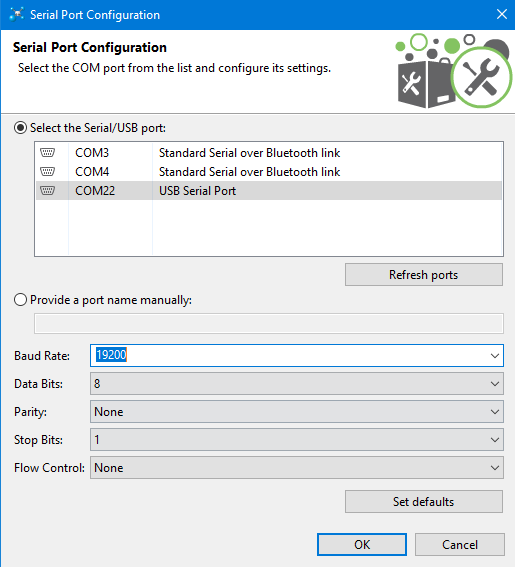


* The serial communication should be established with the following settings:
  + Baud Rate – 9600/19200
  + Data bits – 8
  + Parity – None
  + Flow Control – None
  + Stop bits – 1
  + Data mode – Binary (AVOID USING TEXT MODE.)
* For testing and debugging, the serial console tool in [DiGi XCTU](https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu) software can be used.
* For the ease of use, the command sequences can be exported and imported to the XCTU software, and sequence looping is also supported.
* When using XCTU, make sure that the *hexadecimal view* is enabled through the  button in the XCTU serial console window.
* In XCTU Serial Console, Sent messages are logged in BLUE and Received responses are recorded in RED.

# Initial setup

## Powering up and establishing connection

1. Connect the 18-pin to DB-9 cable to the bill validator and the USB-RS232 converter, and plug the RS-232 converter to the computer.
2. Use ’Device manager’ to obtain the COM port number
3. Open XCTU and goto Serial Console 
4. Click *Configure* and set the settings as follows and click *OK*:



1. Click *Open* to establish the serial connection
2. Connect the 6-pin connector to the bill validator and supply power (+12V) through yellow and black wires
3. Upon powering the unit, the Serial Console will indicate **00** in red colour, and the bill validator’s LED will light up RED.
4. The unit is now ready to accept instructions.

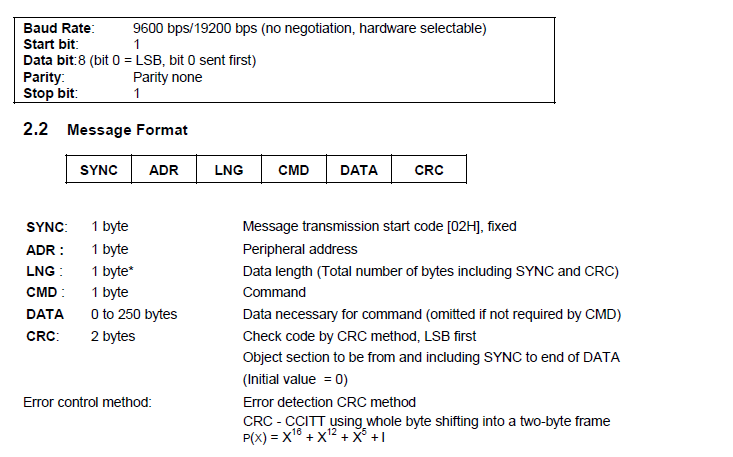
## Communicating with the device

### Important Notes

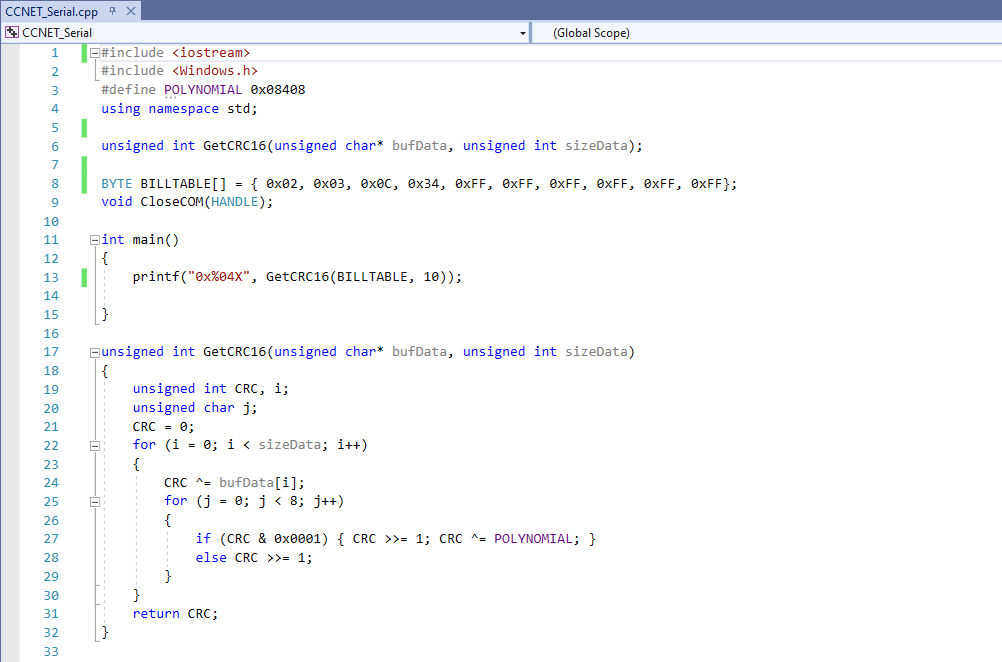
* The Computer/Master Device will be referred as ‘Controller’, and the bill validator will be referred as ‘BV’ here onwards.
* The BV has a unique address of 0x03. (other devices such as coin validators have different addresses to distinguish them)
* The BV only responds when the controller POLLs for information. Otherwise BV stays idle and does not send any information.

### Communication format

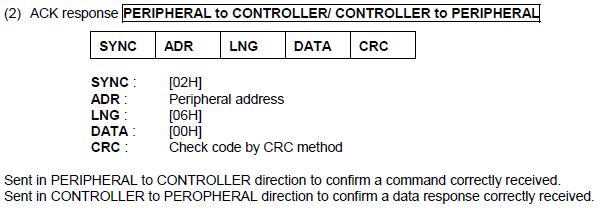
The device only responds to the messages received in the following format:



* The length of the message must exclude the 2-byte CRC16 checksum
* Special care should be taken when preparing the message, since the BV exepects an accurate CRC16 checksum added to the end of the message. The following algorithm can be used to calculate the CRC16 checksum of a given array of hex values. **Make sure that the CRC16 value is appended to the end of the message LSB first format, as shown in the example later in this document.**



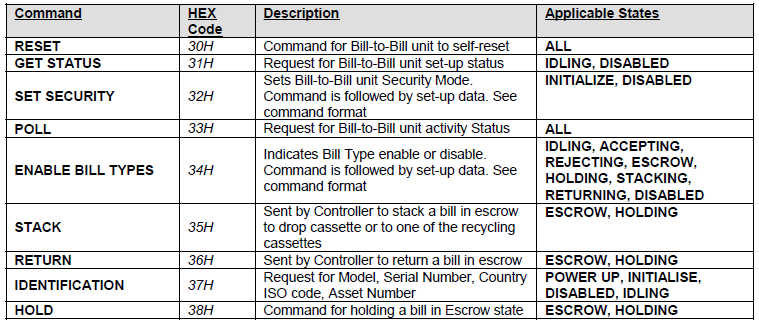
* If a message is successfully received by the BV, it returns an ACK message in the following format:

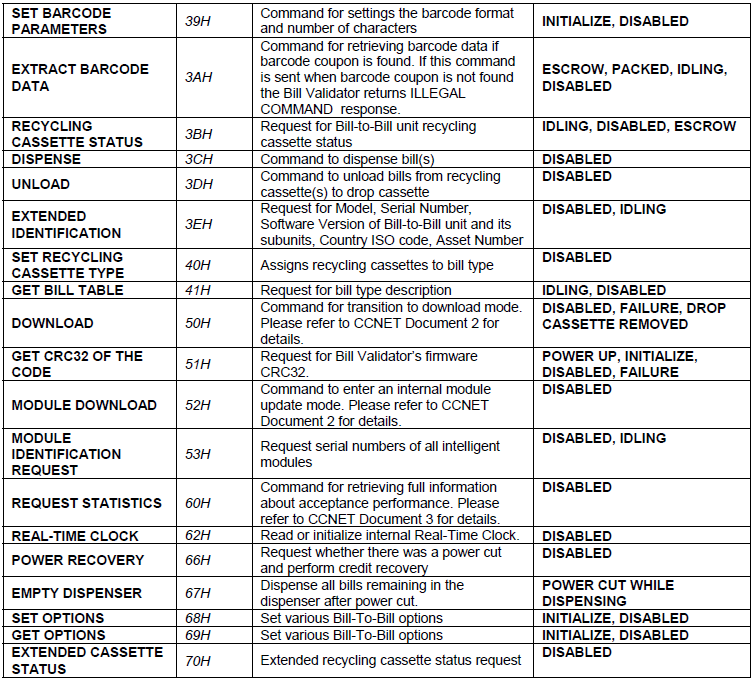


This ACK can be send from controller to BV to acknowledge the received information as well.

* *Refer to pages 9-12 of* [this](https://kiosksoft.ru/device_files/download/87) *document for more information regarding ACK, NACK and INVALID messages.*

### Commands

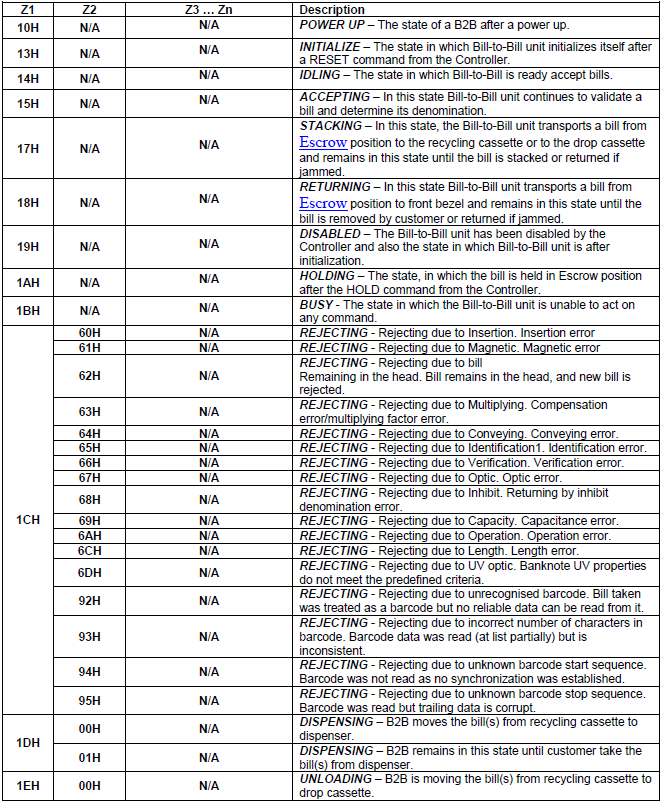


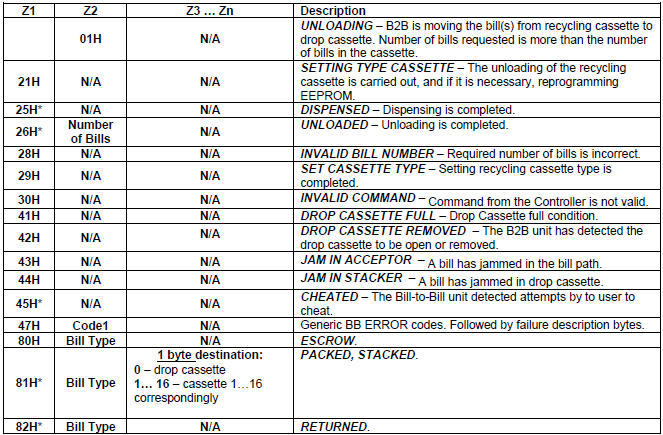


*Refer page 17-22 for detailed reference of the CCNET commands document.*

### POLL Command

This command is frequently sent to BV to obtain the BV unit and its activity. The response usually contains 3 data bytes and depending on the code, may take more bytes. This length can be extracted from the *length* byte.





* POLL command’s response sent by the BV should be acknowledged by the controller by sending ACK after receiving the response.

# Communication Examples

## Example Messages

### POLL

According to the CCNET protocol, the message should take the format as in 2.2.2. Therefore, the POLL message would be:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x02 | 0x03 | 0x06 | 0x33 | 0xDA | 0x81 |

* The red colored two bytes are the CRC16 checksum obtained by calling the function GetCRC16(), up to the number of data bytes.
* For the POLL command, the CRC obtained is 0x81DA. Note how the bytes are swapped when appending to the message.
* LENGTH byte should be **<number of data bytes> + <CRC bytes>**

Example1: The response for trying to identify a deformed bill was received when the bill was fed into the device.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0x02 | 0x03 | 0x07 | 0x1C | 0x66 | 0x41 | 0x85 |

By referring to the table 2.2.4, the data 0x1C 0x66 indicates that it’s a ***REJECTING*** error, with error code 0x66, which indicates that the bill has been rejected due to verification error.

### ACK

* Acknowledgement (Controller to BV or BV to Controller)
* Controller to BV does not return a response. This is normal behavior.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x02 | 0x03 | 0x06 | 0x00 | 0xC2 | 0x82 |

### STATUS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x02 | 0x03 | 0x06 | 0x31 | 0xC8 | 0xA2 |

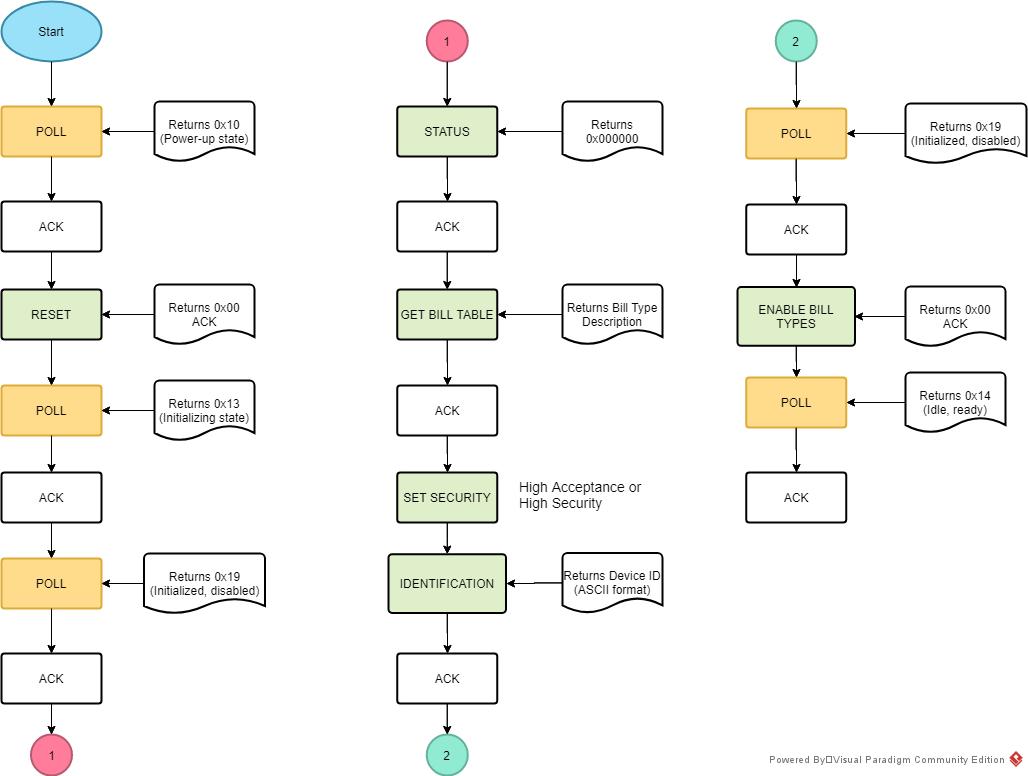
### Disable Bill Accepting

The BV can be disabled by writing 0x000000 to the ENABLE BILL TYPES register

02 03 0C 34 00 00 00 00 00 00 17 0C

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x02 | 0x03 | 0x0C | 0X00 | 0X00 | 0X00 | 0X00 | 0X00 | 0X00 | 0X17 | 0X0C |

## Setup after powering on



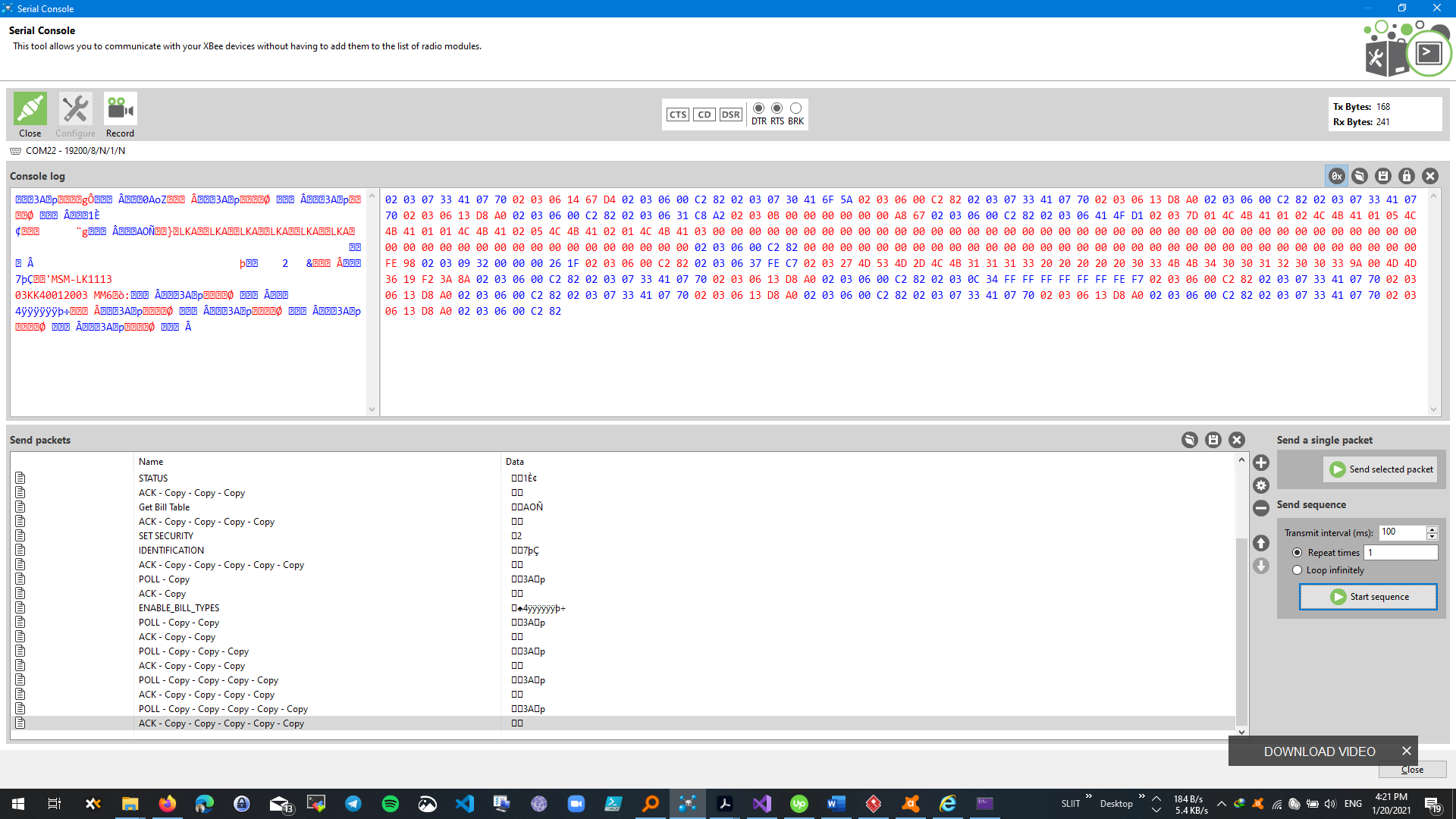
The payload sequence for the above flow diagram is as follows:

|  |  |  |
| --- | --- | --- |
| **Command Name** | **Payload** | **HEX** |
| POLL | 02030733410770 | 0x02,0x03,0x07,0x44,0x41,0x07,0x70 |
| ACK | 02030600C282 | ‘’ |
| RESET | 02030730416F5A | ‘’ |
| POLL | 02030733410770 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| POLL | 02030733410770 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| STATUS | 02030631C8A2 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| GET BILL TABLE | 020306414FD1 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| SET SECURITY | 02030932000000261F | ‘’ |
| GET IDENTIFICATION | 02030637FEC7 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| POLL | 02030733410770 | ‘’ |
| ACK | 02030600C282 | ‘’ |
| ENABLE BILL TYPES | 02030C34FFFFFFFFFFFFFEF7 | ‘’ |
| POLL | 02030733410770 | ‘’ |
| ACK | 02030600C282 | ‘’ |

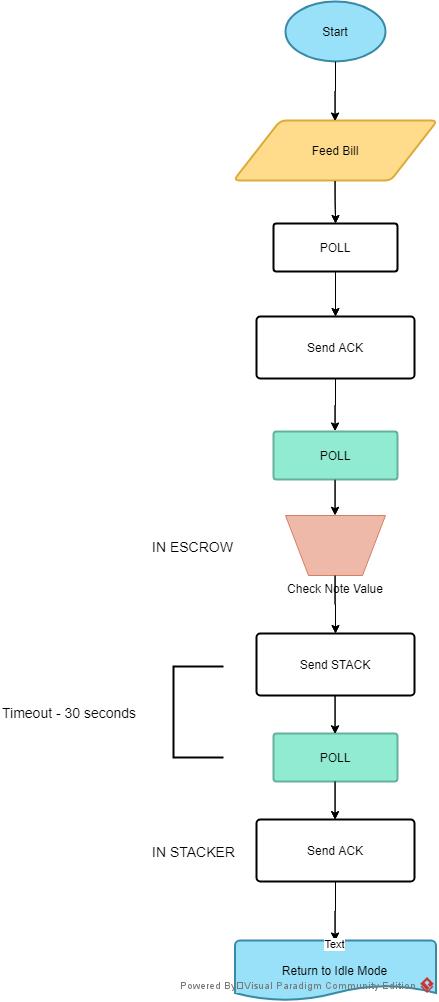
Use the init.xml file to import the above sequence into XCTU and run as sequence once, each line 100ms apart to setup the device in one click.



XCTU output for a single loop



## Bill Accepting Sequence



The controller needs to constantly POLL for new information from the BV in order to detect whether there’s a bill present at the input. When the BV is set to accept bills and in IDLE mode, once the bill has been fed in, the immediate POLL instruction will cause the BV to accept the bill and put it in ESCROWposition.

Upon subsequent POLL requests the BV will return the type of bill in ESCROW and in the meantime, if the bill is detected to be defective due to some error, BV will send a response to the controller after the next POLL request and REJECT the bill.

If the bill is OK and its type is verified, the controller can then issue the STACK command to put the bill into the stacker. STACK command should be followed by a POLL request within 30 seconds and in response, the BV returns a code (0x81 0xNN [where NN is the bill number]) indicating that the stacking sequence is successful. After ensuring the bill is correctly stacked, an ACK command should be send to the BV by the controller to finish the transaction and go back into idling mode. Then the process can repeat to access the next bill, or end accepting bills.

