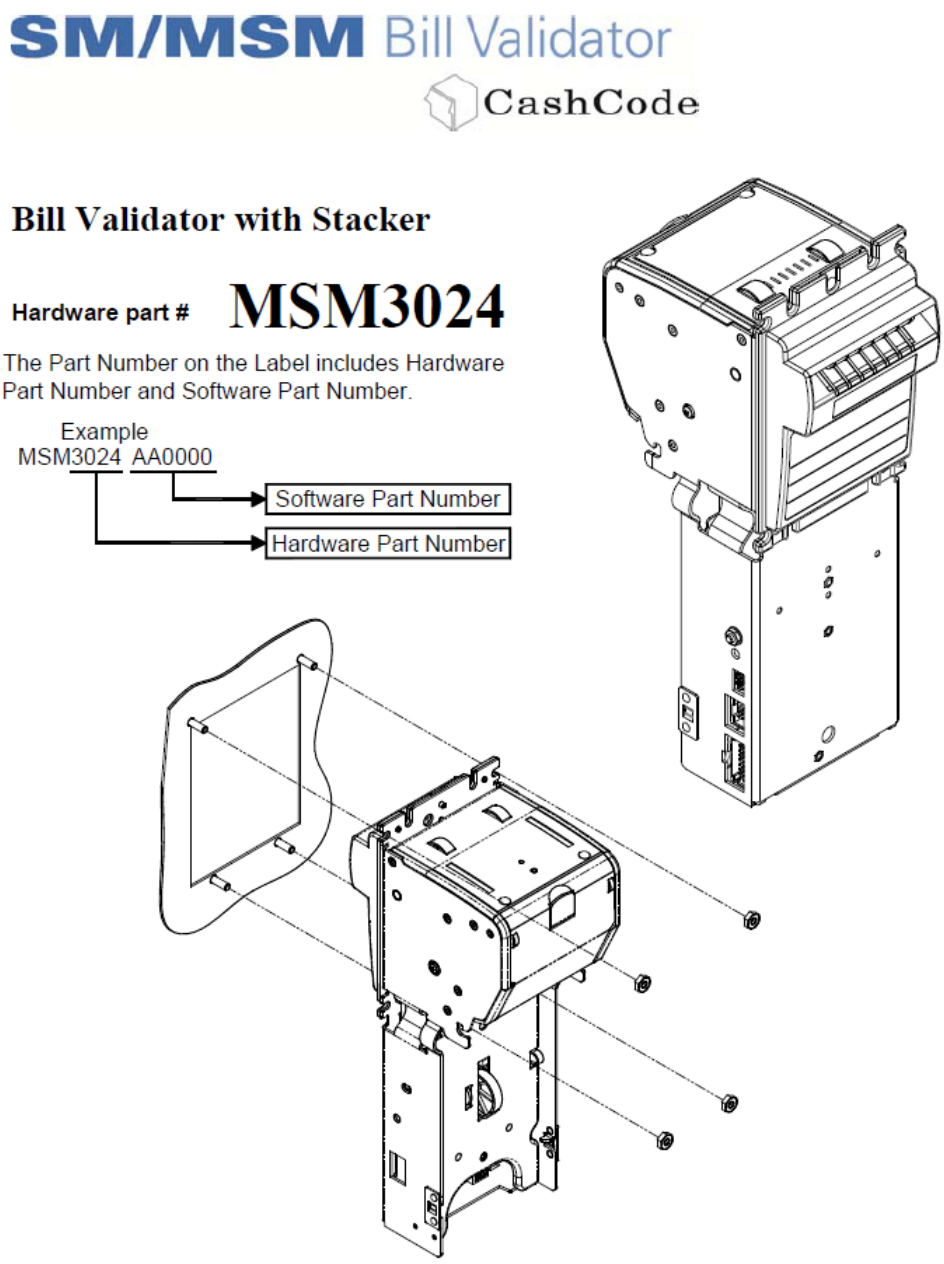
CashCode MSM-3024 Bill Validator

Getting Started Guide



1 Basic Information

1.1 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.

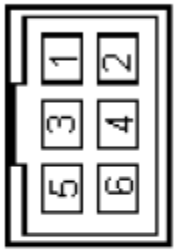
- This unit (firmware LK1113) contains a firmware that can detect and identify Sri Lankan currency from Rs.10 to Rs.1000 notes. The types of notes accepted by the device can be selected through a certain set of commands, and can be overridden by selecting the DIP switches found beneath the stacker dock,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

1.2 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.

6 pin connector pinout

Pin	DESCRIPTION
1	+12V DC
2	GND
3	Pulse Output (NO)
4	Pulse Output Com.
5	Inhibit Line (+)
6	Inhibit Line (-)



Mating connector:

MOLEX: 15-04-5064 -1pcs; 50-57-9303 -2pcs; 16-02-0096 -6pcs;

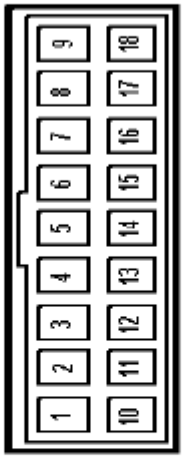
- The unit only requires +12VDC (Yellow) and GND(Black) wires to be powered on. 3, 4, 5, 6 pins can be left floating.

1.3 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 link.

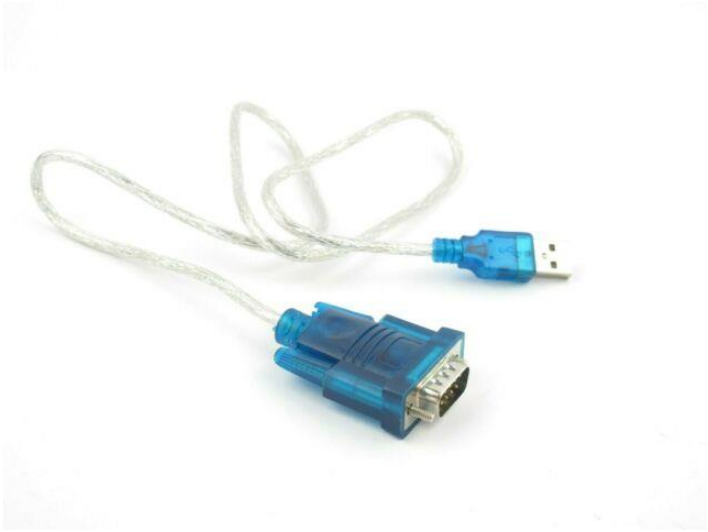
18 pin connector pinout

Pin	DESCRIPTION
1	Credit Pulse
2	Interrupt
3	Serial/Pulse Select
4	Common
5	Serial Data Output
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Out of service
11	Reserved
12	Accept Enable
13	LED Power Source
14	Send
15	RXD
16	Escrow
17	TXD
18	Reserved



Mating connector:
AMP: 102398-7 - 1pcs; 102681-4 - 1pcs; 102536-7 - 1pcs;

- 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.



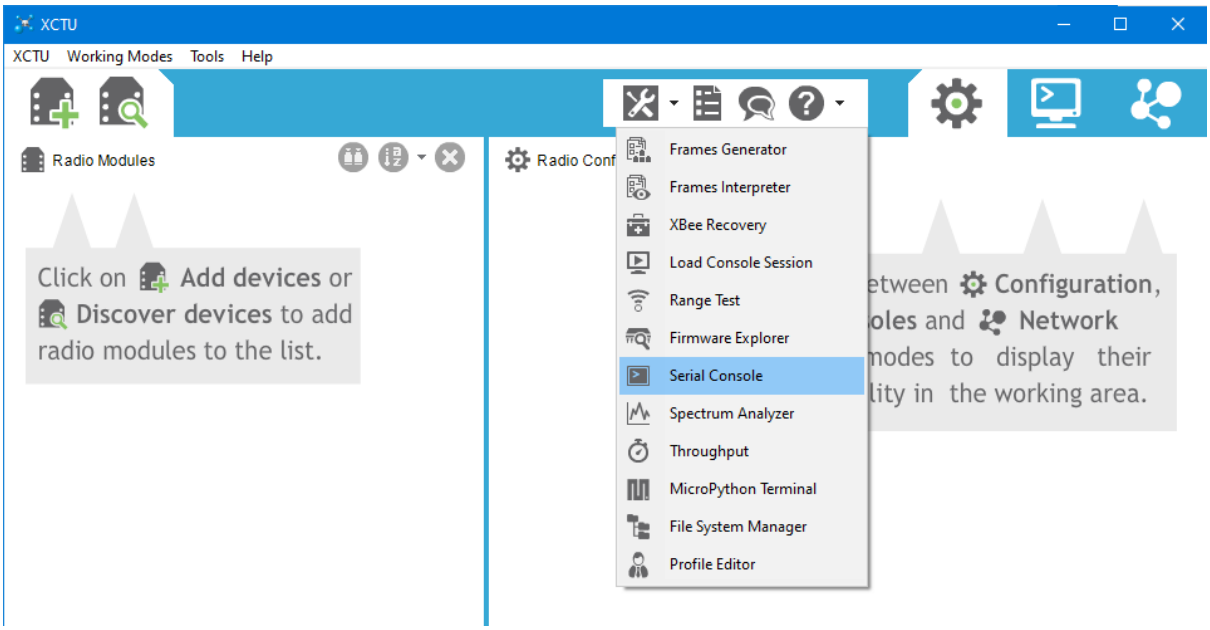
- The serial communication should be established with the following settings:
    - Baud Rate – 9600/19200
    - Data bits – 8
    - Parity – None
    - Flow Control – None
    - Stop bits – 1
    - Data mode – Binary (AVOID USING TEXT MODE.)
- For testing and debugging, the serial console tool in DiGi XCTU 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.

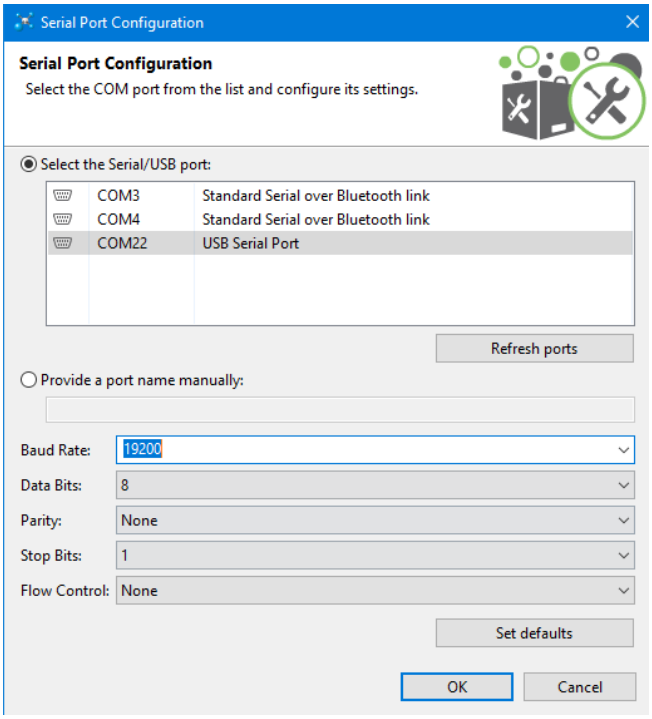
2 Initial setup

2.1 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*:



5. Click *Open* to establish the serial connection

6. Connect the 6-pin connector to the bill validator and supply power (+12V) through yellow and black wires
7. Upon powering the unit, the Serial Console will indicate **00** in red colour, and the bill validator's LED will light up RED.
8. The unit is now ready to accept instructions.

2.2 Communicating with the device

2.2.1 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.

2.2.2 Communication format

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

Baud Rate:	9600 bps/19200 bps (no negotiation, hardware selectable)
Start bit:	1
Data bit:	8 (bit 0 = LSB, bit 0 sent first)
Parity:	Parity none
Stop bit:	1

2.2 Message Format

SYNC	ADR	LNG	CMD	DATA	CRC

SYNC:	1 byte	Message transmission start code [02H], fixed
ADR :	1 byte	Peripheral address
LNG :	1 byte*	Data length (Total number of bytes including SYNC and CRC)
CMD :	1 byte	Command
DATA	0 to 250 bytes	Data necessary for command (omitted if not required by CMD)
CRC:	2 bytes	Check code by CRC method, LSB first
		Object section to be from and including SYNC to end of DATA
		(Initial value = 0)
Error control method:		Error detection CRC method
		CRC - CCITT using whole byte shifting into a two-byte frame
		$P(X) = X^{16} + X^{12} + X^5 + 1$

- 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.**

CCNET_Serial.cpp

CCNET_Serial (Global Scope)

```
1  #include <iostream>
2  #include <Windows.h>
3  #define POLYNOMIAL 0x08408
4  using namespace std;
5
6  unsigned int GetCRC16(unsigned char* bufData, unsigned int sizeData);
7
8  BYTE BILLTABLE[] = { 0x02, 0x03, 0x0C, 0x34, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
9  void CloseCOM(HANDLE);
10
11 int main()
12 {
13     printf("0x%04X", GetCRC16(BILLTABLE, 10));
14
15 }
16
17 unsigned int GetCRC16(unsigned char* bufData, unsigned int sizeData)
18 {
19     unsigned int CRC, i;
20     unsigned char j;
21     CRC = 0;
22     for (i = 0; i < sizeData; i++)
23     {
24         CRC ^= bufData[i];
25         for (j = 0; j < 8; j++)
26         {
27             if (CRC & 0x0001) { CRC >>= 1; CRC ^= POLYNOMIAL; }
28             else CRC >>= 1;
29         }
30     }
31     return CRC;
32 }
33
```

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

(2) ACK response **PERIPHERAL to CONTROLLER/ CONTROLLER to PERIPHERAL**

SYNC	ADR	LNG	DATA	CRC

SYNC : [02H]
ADR : Peripheral address
LNG : [06H]
DATA : [00H]
CRC : Check code by CRC method

Sent in PERIPHERAL to CONTROLLER direction to confirm a command correctly received.
Sent in CONTROLLER to PEROPHERAL direction to confirm a data response correctly received.

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

- *Refer to pages 9-12 of this document for more information regarding ACK, NACK and INVALID messages.*

2.2.3 Commands

Command	HEX Code	Description	Applicable States
RESET	*30H*	Command for Bill-to-Bill unit to self-reset	**ALL**
GET STATUS	*31H*	Request for Bill-to-Bill unit set-up status	**IDLING, DISABLED**
SET SECURITY	*32H*	Sets Bill-to-Bill unit Security Mode. Command is followed by set-up data. See command format	**INITIALIZE, DISABLED**
POLL	*33H*	Request for Bill-to-Bill unit activity Status	**ALL**
ENABLE BILL TYPES	*34H*	Indicates Bill Type enable or disable. Command is followed by set-up data. See command format	**IDLING, ACCEPTING, REJECTING, ESCROW, HOLDING, STACKING, RETURNING, DISABLED**
STACK	*35H*	Sent by Controller to stack a bill in escrow to drop cassette or to one of the recycling cassettes	**ESCROW, HOLDING**
RETURN	*36H*	Sent by Controller to return a bill in escrow	**ESCROW, HOLDING**
IDENTIFICATION	*37H*	Request for Model, Serial Number, Country ISO code, Asset Number	**POWER UP, INITIALISE, DISABLED, IDLING**
HOLD	*38H*	Command for holding a bill in Escrow state	**ESCROW, HOLDING**
SET BARCODE PARAMETERS	*39H*	Command for settings the barcode format and number of characters	**INITIALIZE, DISABLED**
EXTRACT BARCODE DATA	*3AH*	Command for retrieving barcode data if barcode coupon is found. If this command is sent when barcode coupon is not found the Bill Validator returns ILLEGAL COMMAND response.	**ESCROW, PACKED, IDLING, DISABLED**
RECYCLING CASSETTE STATUS	*3BH*	Request for Bill-to-Bill unit recycling cassette status	**IDLING, DISABLED, ESCROW**
DISPENSE	*3CH*	Command to dispense bill(s)	**DISABLED**
UNLOAD	*3DH*	Command to unload bills from recycling cassette(s) to drop cassette	**DISABLED**
EXTENDED IDENTIFICATION	*3EH*	Request for Model, Serial Number, Software Version of Bill-to-Bill unit and its subunits, Country ISO code, Asset Number	**DISABLED, IDLING**
SET RECYCLING CASSETTE TYPE	*40H*	Assigns recycling cassettes to bill type	**DISABLED**
GET BILL TABLE	*41H*	Request for bill type description	**IDLING, DISABLED**
DOWNLOAD	*50H*	Command for transition to download mode. Please refer to CCNET Document 2 for details.	**DISABLED, FAILURE, DROP CASSETTE REMOVED**
GET CRC32 OF THE CODE	*51H*	Request for Bill Validator's firmware CRC32.	**POWER UP, INITIALIZE, DISABLED, FAILURE**
MODULE DOWNLOAD	*52H*	Command to enter an internal module update mode. Please refer to CCNET Document 2 for details.	**DISABLED**
MODULE IDENTIFICATION REQUEST	*53H*	Request serial numbers of all intelligent modules	**DISABLED, IDLING**
REQUEST STATISTICS	*60H*	Command for retrieving full information about acceptance performance. Please refer to CCNET Document 3 for details.	**DISABLED**
REAL-TIME CLOCK	*62H*	Read or initialize internal Real-Time Clock.	**DISABLED**
POWER RECOVERY	*66H*	Request whether there was a power cut and perform credit recovery	**DISABLED**
EMPTY DISPENSER	*67H*	Dispense all bills remaining in the dispenser after power cut.	**POWER CUT WHILE DISPENSING**
SET OPTIONS	*68H*	Set various Bill-To-Bill options	**INITIALIZE, DISABLED**
GET OPTIONS	*69H*	Set various Bill-To-Bill options	**INITIALIZE, DISABLED**
EXTENDED CASSETTE STATUS	*70H*	Extended recycling cassette status request	**DISABLED**

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

2.2.4 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.

Z1	Z2	Z3 ... Zn	Description
10H	N/A	N/A	*POWER UP* – The state of a B2B after a power up.
13H	N/A	N/A	*INITIALIZE* – The state in which Bill-to-Bill unit initializes itself after a RESET command from the Controller.
14H	N/A	N/A	*IDLING* – The state in which Bill-to-Bill is ready accept bills.
15H	N/A	N/A	*ACCEPTING* – In this state Bill-to-Bill unit continues to validate a bill and determine its denomination.
17H	N/A	N/A	*STACKING* – In this state, the Bill-to-Bill unit transports a bill from Escrow position to the recycling cassette or to the drop cassette and remains in this state until the bill is stacked or returned if jammed.
18H	N/A	N/A	*RETURNING* – In this state Bill-to-Bill unit transports a bill from Escrow position to front bezel and remains in this state until the bill is removed by customer or returned if jammed.
19H	N/A	N/A	*DISABLED* – The Bill-to-Bill unit has been disabled by the Controller and also the state in which Bill-to-Bill unit is after initialization.
1AH	N/A	N/A	*HOLDING* – The state, in which the bill is held in Escrow position after the HOLD command from the Controller.
1BH	N/A	N/A	*BUSY* - The state in which the Bill-to-Bill unit is unable to act on any command.
1CH	60H	N/A	*REJECTING* - Rejecting due to Insertion. Insertion error
	61H	N/A	*REJECTING* - Rejecting due to Magnetic. Magnetic error
	62H	N/A	*REJECTING* - Rejecting due to bill Remaining in the head. Bill remains in the head, and new bill is rejected.
	63H	N/A	*REJECTING* - Rejecting due to Multiplying. Compensation error/multiplying factor error.
	64H	N/A	*REJECTING* - Rejecting due to Conveying. Conveying error.
	65H	N/A	*REJECTING* - Rejecting due to Identification1. Identification error.
	66H	N/A	*REJECTING* - Rejecting due to Verification. Verification error.
	67H	N/A	*REJECTING* - Rejecting due to Optic. Optic error.
	68H	N/A	*REJECTING* - Rejecting due to Inhibit. Returning by inhibit denomination error.
	69H	N/A	*REJECTING* - Rejecting due to Capacity. Capacitance error.
	6AH	N/A	*REJECTING* - Rejecting due to Operation. Operation error.
	6CH	N/A	*REJECTING* - Rejecting due to Length. Length error.
	6DH	N/A	*REJECTING* - Rejecting due to UV optic. Banknote UV properties do not meet the predefined criteria.
	92H	N/A	*REJECTING* - Rejecting due to unrecognised barcode. Bill taken was treated as a barcode but no reliable data can be read from it.
	93H	N/A	*REJECTING* - Rejecting due to incorrect number of characters in barcode. Barcode data was read (at list partially) but is inconsistent.
	94H	N/A	*REJECTING* - Rejecting due to unknown barcode start sequence. Barcode was not read as no synchronization was established.
	95H	N/A	*REJECTING* - Rejecting due to unknown barcode stop sequence. Barcode was read but trailing data is corrupt.
1DH	00H	N/A	*DISPENSING* – B2B moves the bill(s) from recycling cassette to dispenser.
	01H	N/A	*DISPENSING* – B2B remains in this state until customer take the bill(s) from dispenser.
1EH	00H	N/A	*UNLOADING* – B2B is moving the bill(s) from recycling cassette to drop cassette.

Z1	Z2	Z3 ... Zn	Description
	01H	N/A	*UNLOADING* – B2B is moving the bill(s) from recycling cassette to drop cassette. Number of bills requested is more than the number of bills in the cassette.
21H	N/A	N/A	*SETTING TYPE CASSETTE* – The unloading of the recycling cassette is carried out, and if it is necessary, reprogramming EEPROM.
25H*	N/A	N/A	*DISPENSED* – Dispensing is completed.
26H*	Number of Bills	N/A	*UNLOADED* – Unloading is completed.
28H	N/A	N/A	*INVALID BILL NUMBER* – Required number of bills is incorrect.
29H	N/A	N/A	*SET CASSETTE TYPE* – Setting recycling cassette type is completed.
30H	N/A	N/A	*INVALID COMMAND* – Command from the Controller is not valid.
41H	N/A	N/A	*DROP CASSETTE FULL* – Drop Cassette full condition.
42H	N/A	N/A	*DROP CASSETTE REMOVED* – The B2B unit has detected the drop cassette to be open or removed.
43H	N/A	N/A	*JAM IN ACCEPTOR* – A bill has jammed in the bill path.
44H	N/A	N/A	*JAM IN STACKER* – A bill has jammed in drop cassette.
45H*	N/A	N/A	*CHEATED* – The Bill-to-Bill unit detected attempts by to user to cheat.
47H	Code1	N/A	Generic BB ERROR codes. Followed by failure description bytes.
80H	Bill Type	N/A	*ESCROW*.
81H*	Bill Type	**1 byte destination:** **0** – drop cassette **1... 16** – cassette 1...16 correspondingly	*PACKED, STACKED*.
82H*	Bill Type	N/A	*RETURNED*.

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

3 Communication Examples

3.1 Example Messages

3.1.1 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.

3.1.2 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

3.1.3 STATUS

0x02	0x03	0x06	0x31	0xC8	0xA2

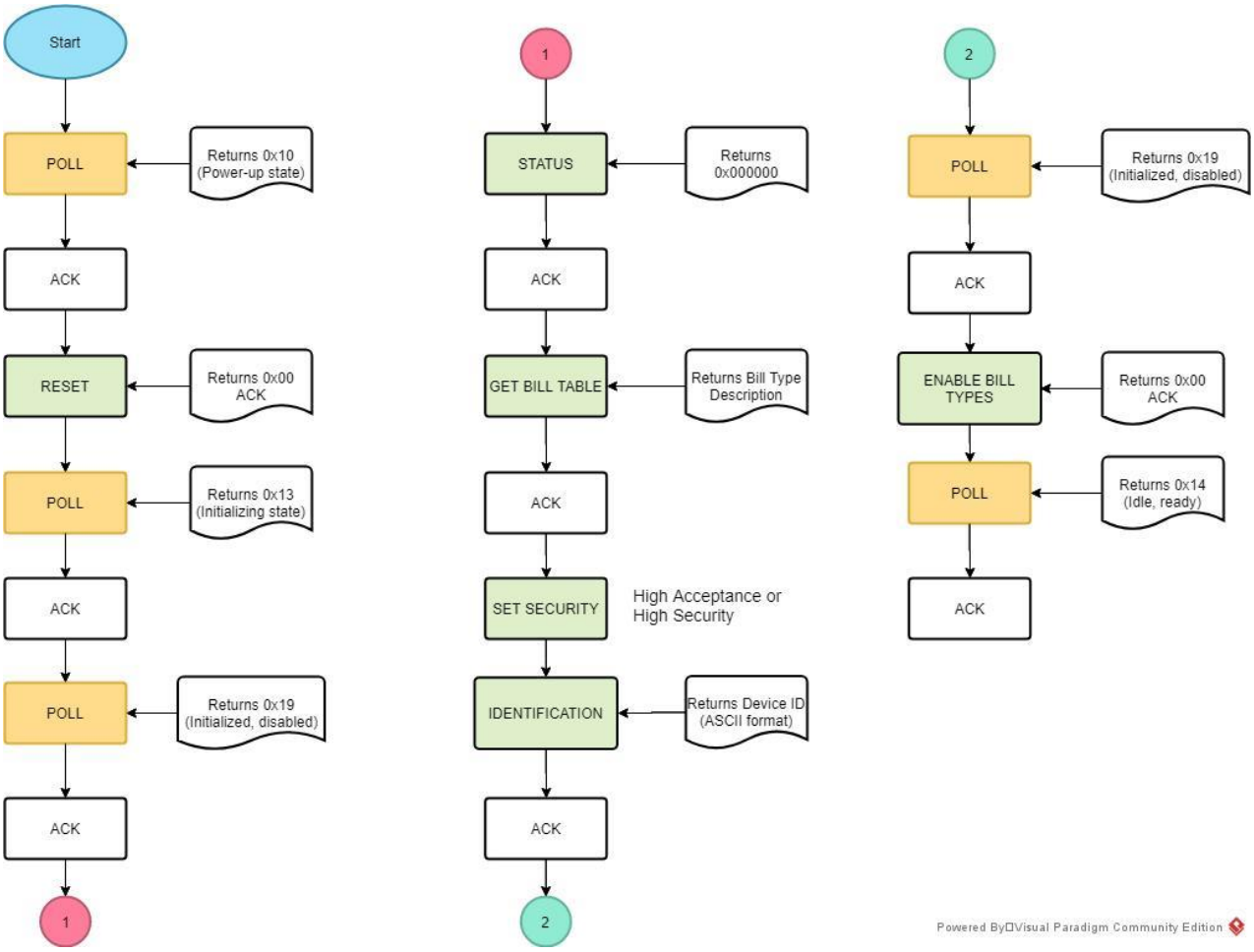
3.1.4 Disable Bill Accepting

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

3.2 Setup after powering on



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

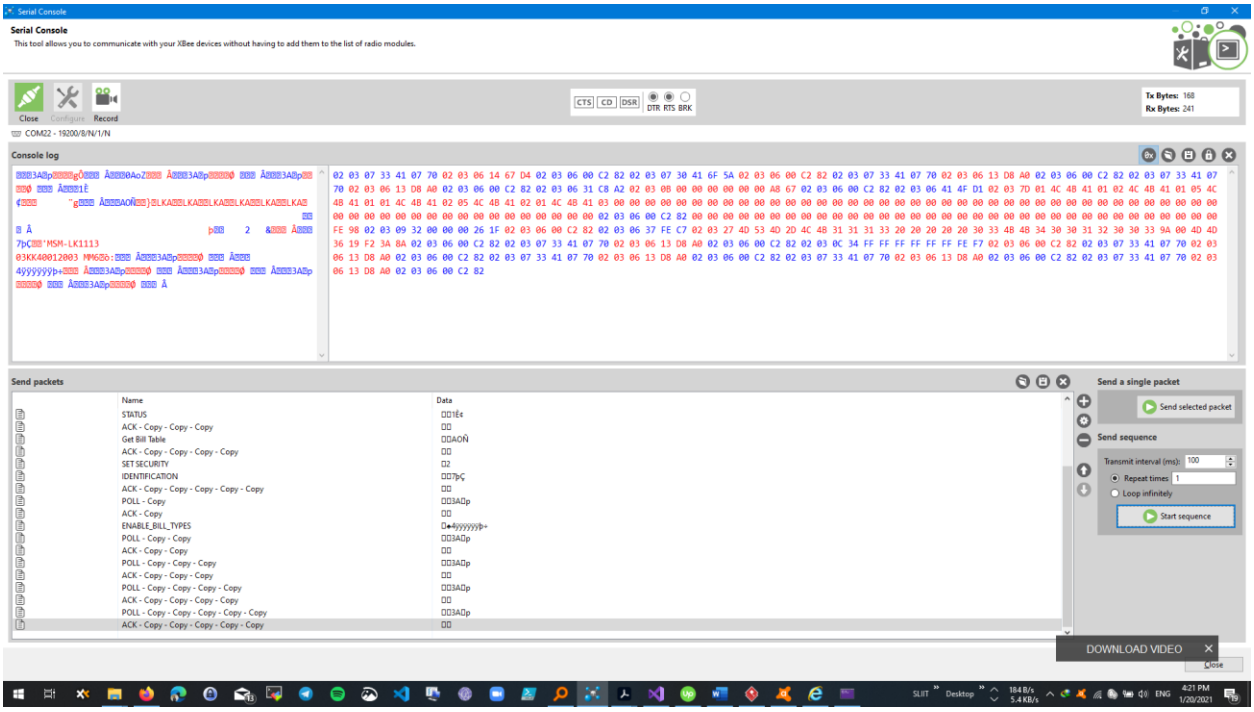
Command Name	Payload	HEX
POLL	02030733410770	0x02,0x03,0x07,0x44,0x41,0x07,0x70
ACK	02030600C282	”
RESET	02030730416F5A	”
POLL	02030733410770	”
ACK	02030600C282	”
POLL	02030733410770	”
ACK	02030600C282	”
STATUS	02030631C8A2	”
ACK	02030600C282	”
GET BILL TABLE	020306414FD1	”
ACK	02030600C282	”

SET SECURITY	02030932000000261F	”
GET IDENTIFICATION	02030637FEC7	”
ACK	02030600C282	”
POLL	02030733410770	”
ACK	02030600C282	”
ENABLE BILL TYPES	02030C34FFFFFFFFFFFFFEF7	”
POLL	02030733410770	”
ACK	02030600C282	”

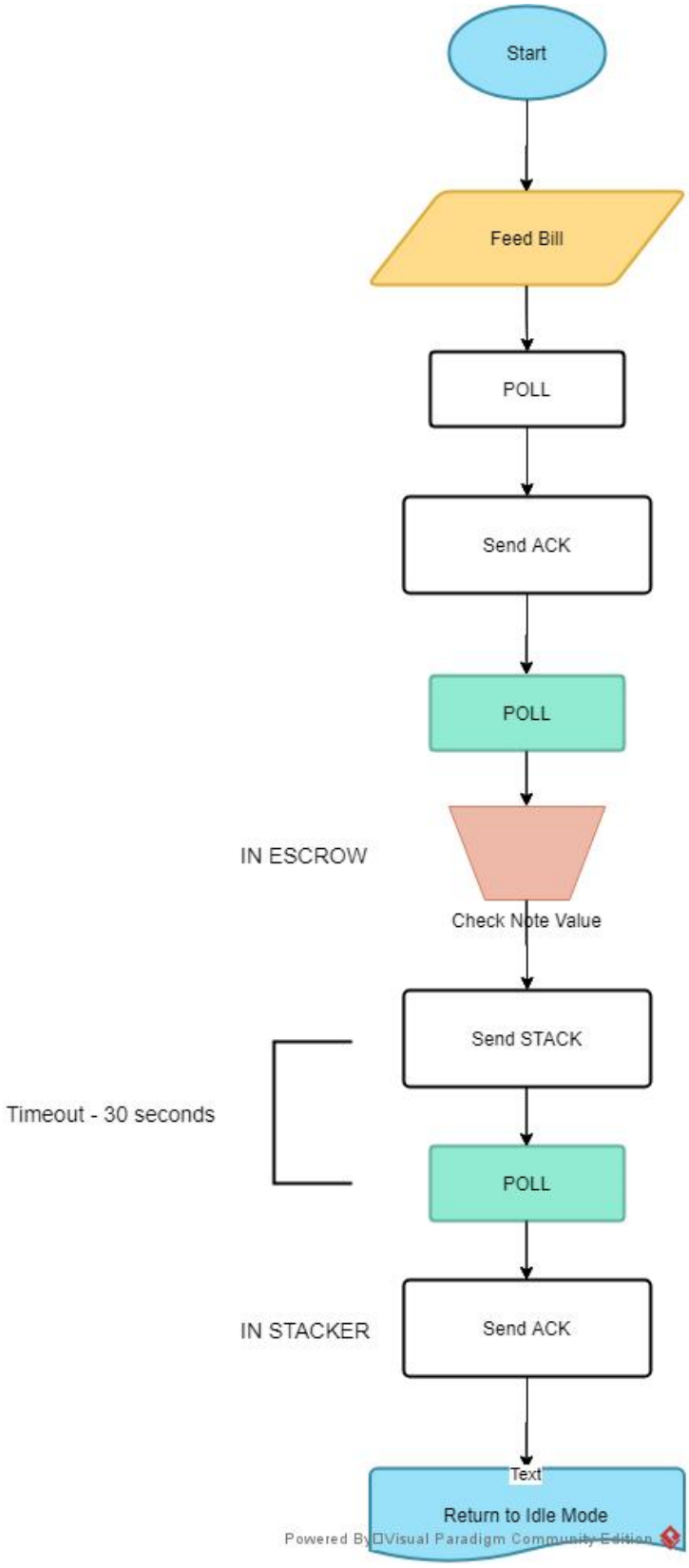
Use the init.xml file to import the above sequence into XCTU and run as sequence once, each line 100ms apart to setup the device in one click.

init.xml

XCTU output for a single loop



3.3 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 ESCROW position.

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.