PICOVEND BLUETOOTH MDB MASTER

v1.0 – 15.02.2019

Table of Contents

I. Hardware description

The device was designed for in-field MDB communication and payment systems testing and diagnostic. It needs external 24VDC power supply to work and it is covering the following functions:

- receiving 9bit MDB communication messages and converting them to 8bit messages;
- handling ACK/NAK in a timely manner (following MDB timings restrictions);
- handling MDB messages in a timely manner (following MDB timings restrictions);

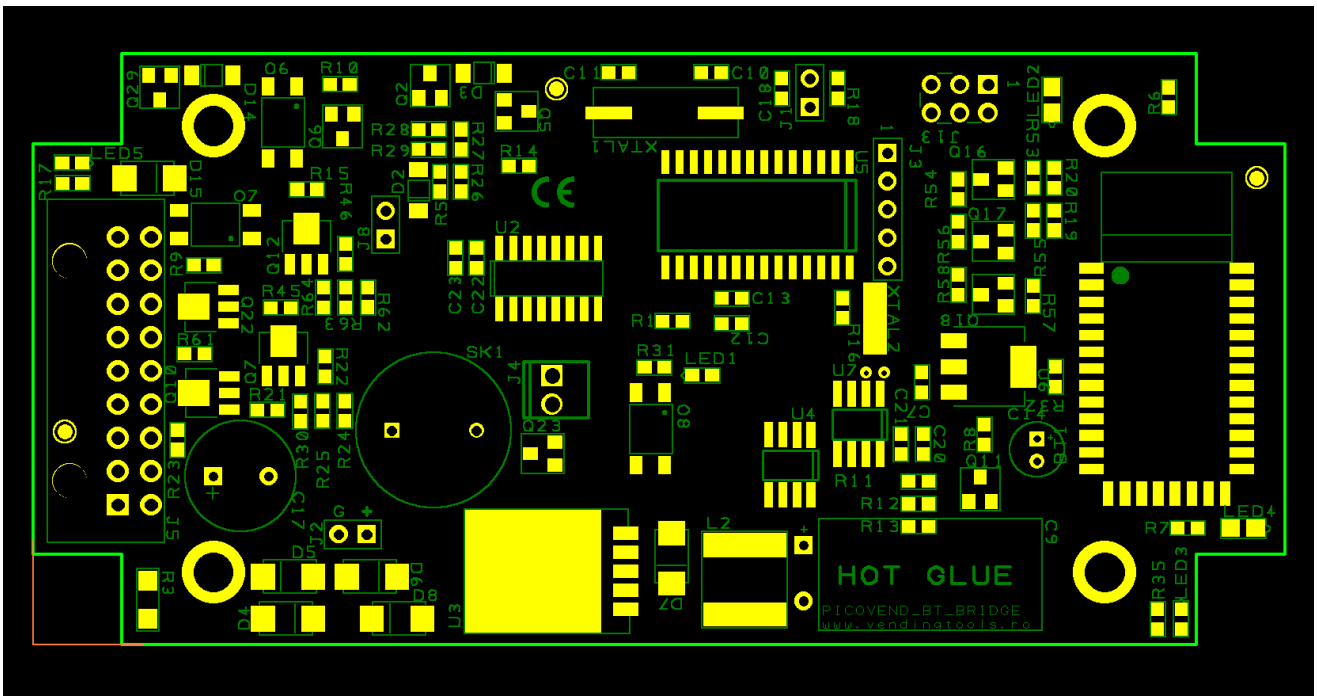

Image 1: Board overview

II. Functionality

The device can be used to handle MDB payment systems from a computer, a SBC (like Raspberry Pi, Orange Pi, etc.) or a phone/tablet. We are offering an Android application that can be used in-field to quickly test MDB peripherals.

A. Connecting to the device using Microsoft Windows 10 operating system

To connect the device on a computer using Microsoft Windows 10 operating system, you need to follow the procedure:

1. Pairing the device over Bluetooth

- Power up the device, applying a voltage between 10 and 34VDC, depending on your MDB peripherals working voltage.
- Go to “Bluetooth and other devices setting”
- Choose “Add Bluetooth or other device”
- Choose “Bluetooth (mice, keyboard, pens, or audio and other kinds of Bluetooth device)”

- Wait for the the operating system to detect and identify the device. Sniffers Bluetooth IDs are respecting the following format: PVBTAAXXXX, where AA is the device family and XXXX is the device ID (both can be letters and numbers).
- Select the device, type the password on the device's label and wait for the operating system to install needed file (detection and installation are automatically performed, no software installation needed).
- After the finish message installation, check if the device is correctly installed (right click on "This PC" → click on "Manage" → click on "Device manager" → expand "Ports (COM & LPT) and search for one or more records "Standard Serial over Bluetooth link (COMx or COMxx).

2. Using the device over Bluetooth

You can use some script capable serial communication application, or develop your own application, following low level device protocol handling.

B. Connecting to the device using Ubuntu or Raspbian (for Raspberry Pi)

- Power up the device, applying a voltage between 10 and 34VDC, depending on your MDB peripherals working voltage.
- Open a terminal window and change user to root.
- Using your favorite editor, modify the file /etc/systemd/system/dbus-org.bluez.service as follows:
    - You need to have a line containing - ExecStart=/usr/lib/bluetooth/bluetoothd -C
    - You need to have a line containing - ExecStartPost=/usr/bin/sdptool add SP
- Save and reboot
- Open a terminal window and change user to root.
- Run the command bluetoothctl
- In bluetoothctl console, type the following command succession:
    - scan on
    - devices
    - agent on
    - pair XX:XX:XX:XX:XX:XX (where XX:XX:XX:XX:XX:XX is the hardware address of your device)
    - type the password you can find on device's label when asked to.
    - trust XX:XX:XX:XX:XX:XX
    - quit

In the console, type: rfcomm bind /dev/rfcomm0 XX:XX:XX:XX:XX 1
After that, you can use any serial terminal application to connect on /dev/rfcomm0
Please make sure you configure the serial port with 115200bps, 8 data bits, 1 stop bit, no parity and no hardware or software flow control. Also, please make sure you are selecting HEX data view.

III. The low level protocol over Bluetooth

The low level protocol is the most convenient and flexible interface working mode, since it allows the developers to send virtually any command to the MDB bus.

The device is sending periodically a poll over the Bluetooth and after receiving the poll, the user application has 500ms to answer with a command for MDB bus. If the user application is not sending any message, then the device will send another poll message over the Bluetooth.

The poll message from the device is fixed length, with fixed content, as follows:
- 0xF0 0x50 0x53

Your application should wait for this message, then send any needed MDB command.
There are, also some reserved commands, for device settings and registers manipulation.
ATENTION!!! - Any command sent later than 500ms will be ignored or will cause a NAK response from the device. You need to keep the commands prepared and send immediately after poll message received.

A. Reserved device commands

Any reserved command starts with a 0xFE byte
The device is answering to any command with a command ACK or NAK as follows:
- 0xFC 0xFC 0xFC 0xFC 0xF0 – ACK
- 0xFD 0xFD 0xFD 0xFD 0xF0 – NAK

Last byte on commands and answers is the CRC and is calculated by the same procedure used for MDB.

1. Clear paired devices

This command clears all previous paired devices saved into the device. You will need to pair it again with your needed devices. The device is supporting a maximum of 6 paired devices so, from time to time it is necessary to clear paired devices to allow it be paired with others.

Command from user app	
0xFE 0x03 <CRC>	Clear paired devices
Device answer	
- 0xFC 0xFC 0xFC 0xFC 0xF0 – ACK - 0xFD 0xFD 0xFD 0xFD 0xF0 – NAK	

2. Press a key on optional keyboard simulator board

This command will force a key press (working only with optional keyboard simulator board)

Command from user app	
0xFE 0x04 <ROW> <COL> <T1> <T0> <CRC>	<ROW> is the row relay to energize <COL> is the column relay to energize <T1> most significant byte for delay after energizing (ms) <T0> less significant byte for delay after energizing (ms)
Device answer	
- 0xFC 0xFC 0xFC 0xFC 0xF0 – ACK - 0xFD 0xFD 0xFD 0xFD 0xF0 – NAK	ACK comes after the delay

B. MDB commands

Any other message your application is sending to the device as a response to the device poll, will be interpreted as a message that should be send to the MDB bus. It is the responsibility of your application to create correct MDB message (including the correct CRC).

We are offering a demo application (daemon) that can be used for high level protocol (described in a different section below). The demo application is developed using B4J (https://www.b4x.com/b4j.html), a simple and intuitive programming language, with a completely free RAD IDE and we are providing the source code in the download archive. The demo is a console application and was tested on Windows 10, Ubuntu and Raspbian (Raspberry Pi), it is a JAR application and by creating a script you can run it as a service.

IV. The high level protocol over Bluetooth, using our demo application

We are offering a demo application (daemon) that can be used for high level protocol (described in a different section below). The demo application is developed using B4J (https://www.b4x.com/b4j.html), a simple and intuitive programming language, with a completely free RAD IDE and we are providing the source code in the download archive. The demo is a console application and was tested on Windows 10, Ubuntu and Raspbian (Raspberry Pi), it is a JAR application and by creating a script you can run it as a service.
In high level protocol, the communication is based on sending and receiving JSON messages on an open socket.
To install our demo application, make sure you have Java 8 SDK installed on your computer, then copy the demo application in a dedicated folder and launch it with the following command:

#java -jar pvmdbmstd.jar

Then connect your your application on a socket port 5131 TCP and start sending high level protocol commands. For testing purposes, you can even use TELNET or an easiest way, by installing Packet Sender (available for Windows, GNU/Linux, Mac OS, Android and iOS at https://packetsender.com/)
We are also providing some Packet Sender exports for download, where you need to change the target IP to match your computer/Raspberry Pi address.
If you are turning off interface's power, the demo app will disconnect from Bluetooth and this action will generate a, exception outside of Java virtual machine, that will shut down the application. This exception cannot be catched by the demo app so, it is recommended to start the application using a script that will keep relaunching it with a few seconds delay. Also, you need to monitor the socket status in your application and reopen it accordingly when it becomes available again.
JSON messages you can send to the demo app are, of course, case sensitive. There are 2 JSON types, commands and messages. Commands are those that are modifying demo app behavior and settings on the fly, while messages are those that will be sent directly to the MDB bus. If the commands are not correctly received or interpreted by the demo app, there will be no answer. Otherwise, the application respond with a CMDConfirm for MessageType.
For example: {"Value":"CMDBillAutoPollOn","MessageType":"CMDConfirm"}
Also, for commands destinated to the device, it will answer with
- {"NumberValue":[252,252,252,252,240],
"StringHexValue":"FCFCFCFCF0","MessageType":"MDBMessage"} for ACK

and with
- {"NumberValue":[253,253,253,253,240],
"StringHexValue":"FDFDFDFDF0","MessageType":"MDBMessage"} for NAK

All responses that could not be parsed by the demo app will trigger a JSON response that is including the received bytes, both as number an hex string.
Ex. {"NumberValue":[240,81,84],"StringHexValue":"F05154","MessageType":"MDBMessage"}

MDB answers on peripheral polls, are coming in the same structure and your application must parse them, for example:
- {"NumberValue":[6,6],"StringHexValue":"0606","MessageType":"MDBMessage"} means that the bill validator was reset. Your application should keep tracking of the last message sent to the MDB peripheral and parse the response accordingly. There are some messages that are returning interpreted results and they will be detailed in the messages section.

A. Commands

1. Set bill peripheral auto-poll ON

This command will set the application in a bill auto-poll mode so, you will not be forced to implement the poll in your application.

Command from user app	
{"Command":"BillAutoPollOn"}	Turns ON the bill peripheral auto-poll. You will also need to supply the Bill connect command, described below.
Device answer	
{"Value":"CMDBillAutoPollOn","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

2. Set bill peripheral auto-poll OFF

This command will turn off Bill auto-poll mode so, you will be forced to implement the poll in your application.

Command from user app	
{"Command":"BillAutoPollOff"}	Turns OFF the bill peripheral auto-poll. You will also need to supply the Bill connect command, described below.
Device answer	
{"Value":"CMDBillAutoPollOff","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

3. Set bill connect

After this command, the demo app will consider that there is a bill validator connected on MDB bus.

Command from user app	
{"Command":"BillConnect"}	The demo app will consider that there is a bill peripheral connected to the MDB bus
Device answer	
{"Value":"CMDBillConnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

4. Set bill disconnect

After this command, the demo app will consider that there is no bill validator connected on MDB bus.

Command from user app	
{"Command":"BillDisconnect"}	The demo app will consider that there is no bill peripheral connected to the MDB bus
Device answer	
{"Value":"CMDBillDisconnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

5. Set coin peripheral auto-poll ON

This command will set the application in a coin auto-poll mode so, you will not be forced to implement the poll in your application.

Command from user app	
{"Command":"CoinAutoPollOn"}	Turns ON the bill peripheral auto-poll. You will also need to supply the coin connect command, described below.
Device answer	
{"Value":"CMDCoinAutoPollOn","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

6. Set coin peripheral auto-poll OFF

This command will turn off coin auto-poll mode so, you will be forced to implement the poll in your application.

Command from user app	
{"Command":"CoinAutoPollOff"}	Turns OFF the coin peripheral auto-poll. You will also need to supply the coin connect command, described below.
Device answer	
{"Value":"CMDCoinAutoPollOff","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

7. Set coin connect

After this command, the demo app will consider that there is a coin acceptor connected on MDB bus.

Command from user app	
{"Command":"CoinConnect"}	The demo app will consider that there is a coin peripheral connected to the MDB bus
Device answer	
{"Value":"CMDCoinConnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

8. Set bill disconnect

After this command, the demo app will consider that there is no coin acceptor connected on MDB bus.

Command from user app	
{"Command":"CoinDisconnect"}	The demo app will consider that there is no bill peripheral connected to the MDB bus
Device answer	
{"Value":"CMDCoinDisconnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

9. Set cashless peripheral auto-poll ON

This command will set the application in a cashless auto-poll mode so, you will not be forced to implement the poll in your application.

Command from user app	
{"Command":"CashlessXAutoPollOn"}	Turns ON the cashless peripheral auto-poll. You will also need to supply the cashless connect command, described below. “X” can be 1 for the first cashless device (poll address 0x12) or 2 for the second cashless device (poll address 0x62)
Device answer	
{"Value":"CMDCashlessXAutoPollOn", "MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

10. Set cashless peripheral auto-poll OFF

This command will turn off cashless auto-poll mode so, you will be forced to implement the poll in your application.

Command from user app	
{"Command":"CashlessXAutoPollOff"}	Turns OFF the peripheral auto-poll. You will also need to supply the cashless connect command, described below. “X” can be 1 for the first cashless device (poll address 0x12) or 2 for the second cashless device (poll address 0x62)
Device answer	
{"Value":"CMDCashlessXAutoPollOff", "MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

11. Set cashless connect

After this command, the demo app will consider that there is a cashless peripheral connected on MDB bus.

Command from user app	
{"Command":"CashlessXConnect"}	The demo app will consider that there is a cashless peripheral connected to the MDB bus. “X” can be 1 for the first cashless device (poll address 0x12) or 2 for the second cashless device (poll address 0x62)
Device answer	
{"Value":"CMDCashlessXConnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

12. Set bill disconnect

After this command, the demo app will consider that there is no cashless peripheral connected on MDB bus.

Command from user app	
{"Command":"CashlessXDisconnect"}	The demo app will consider that there is no cashless peripheral connected to the MDB bus
Device answer	
{"Value":"CMDCashlessXDisconnected","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

13. Reset all paired devices

Command from user app	
{"Command":"ClearPairedDevices"}	This command will reset all paired devices from interface's memory. After issuing this command, you need to follow the Bluetooth pairing procedure again. It will force a Bluetooth disconnect and the demo application stops.
Device answer	
{"Value":"ClearPairedDevice","MessageType":"CMDConfirm"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

14. Set RTC

Command from user app	
{"Command":"SetRTC","Hour":20,"Minutes":28,"Seconds":00,"Day":3,"Month":12,"Year":18,"DayOfWeek":1}	This command will set device's internal RTC
Device answer	
{{"Value":"CMDSetRTC","MessageType":"CMDConfirm"}, also, this message is followed by the device's answer {"NumberValue":[252,252,252,252,240], "StringHexValue":"FCFCFCFCF0", "MessageType":"MDBMessage"} – for ACK or {"NumberValue":[253,253,253,253,240], "StringHexValue":"FDFDFDFDF0", "MessageType":"MDBMessage"} – for NAK	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer.

15. Get RTC

Command from user app	
{"Command":"GetRTC"}	This command will get device's internal RTC
Device answer	
{{"Value":"CMDGetRTC","MessageType":"CMDConfirm"}, also, this message is followed by the device's answer {"NumberValue":[254,2,70,49,32,1,3,18,24,197], "StringHexValue":"FE02463120010312 18C5", "MessageType":"MDBMessage"}	Confirmation answer from the demo app. If the command was not received or correctly interpreted by the demo app, there will be no answer. B0 – always 0xFE (254 decimal) B1 – always 0x02 (2 decimal) B2-B8 – date-time register, according to RTC chip documentation (DS1307), always in 24h format. B9 - <CRC>

B. Messages

1. Bill peripheral reset

Command from user app	
{"Message":"BillReset"}	This command will send bill reset reset command on the MDB bus
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description.

2. Bill poll

Command from user app	
{"Message":"BillPoll"}	This command will send bill poll command on the MDB bus
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a specific peripheral message, for example: {"NumberValue":[6,6], "StringHexValue":"0606", "MessageType":"MDBMessage"} – (in this case “BILL WAS RESET”)	Confirmation answer from the demo app/device, according to MDB protocol description

3. Bill get setup

Command from user app	
{"Message":"BillGetSetup"}	This command will send bill setup command on the MDB bus
Device answer	
{"NumberValue": [1,22,66,0,100,2,1,44,255,255,255,1,5,10,0,0,0,0,0,0,0,0,0,0,0,0,0,249], "StringHexValue":"011642006402012CFFFFFF01050A00000000000000000000000000F9", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

4. Bill expansion identification

Command from user app	
{"Message":"BillExpansionIdentification"}	This command will send bill expansion identification command on the MDB bus
Device answer	
{"NumberValue": [73,84,76,48,48,48,48,53,49,48,48,53,53,54,57,66,86,48,48,50,48,32,32,32,48,48,48,4,18,168], "StringHexValue":"49544C303030303531303035353639425630303230202020303030412A8", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

5. Bill expansion identification with option bits

Command from user app	
{"Message":"BillExpansionIdentificationOption"}	This command will send bill expansion identification w/ option bits command on the MDB bus
Device answer	
{"NumberValue": [73,84,76,48,48,48,48,48,48,50,55,49,50,54,57,66,86,48,49,48,48,32,32,32,48,48,48,4,20,0,0,0,0,165], "StringHexValue":"49544C3030303030303237313236394256303130302020303030041400000000A5", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

6. Bill expansion feature enable

Command from user app	
{"Message":"BillExpansionFeatureEnable","FTL":0,"BillRecycling":1}	This command will send bill expansion feature enable command on the MDB bus
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

7. Bill stacker

Command from user app	
{"Message":"BillStacker"}	This command will send bill stacker command on the MDB bus
Device answer	
{"NumberValue":[0,0,0], "StringHexValue":"000000", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

8. Bill enable

Command from user app	
{"Message":"BillEnable"}	This command will send bill enable command on the MDB bus
Device answer	
No specific answer, soon the bill peripheral will start answering ACK on bill polls	

9. Bill disable

Command from user app	
{"Message":"BillDisable"}	This command will send bill disable command on the MDB bus
Device answer	
No specific answer, soon the bill peripheral will start answering “DISABLED” on bill polls	

10. Bill accept

Command from user app	
{"Message":"BillAccept"}	This command will send bill accept command on the MDB bus. You may issue this message only if you activated bill escrow function and if the bill peripheral has previously responded with a “BILL IN ESCROW” status on the bill poll message
Device answer	
No specific answer, soon the bill peripheral will answer with a bill related messaje (stacked, rejected, etc.) to a bill poll message	

11. Bill reject

Command from user app	
{"Message":"BillReject"}	This command will send bill reject command on the MDB bus. You may issue this message only if you activated bill escrow function and if the bill peripheral has previously responded with a “BILL IN ESCROW” status on the bill poll message
Device answer	
No specific answer, soon the bill peripheral will answer with a bill related messaje (stacked, rejected, etc.) to a bill poll message	

12. Coin reset

Command from user app	
{"Message":"CoinReset"}	This command will send coin reset command
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

13. Coin poll

Command from user app	
{"Message":"CoinReset"}	This command will send coin reset command
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a coin status message, according to MDB specifications	Confirmation answer from the demo app/device, according to MDB protocol description

14. Coin get setup

Command from user app	
{"Message":"CoinGetSetup"}	This command will send coin setup command
Device answer	
Returns coin setup information, for example: {"NumberValue": [3,22,66,10,2,0,3,1,5,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,127], "StringHexValue":"0316420A02000301050F000000000000000000000000007F",	Confirmation answer from the demo app/device, according to MDB protocol description

"MessageType":"MDBMessage"}	

15. Coin expansion identification

Command from user app	
{"Message":"CoinExpansionIdentification"}	This command will send coin expansion identification command to the MDB bus.
Device answer	
Returns coin expansion identification data, for example: {"NumberValue": [77,69,73,52,50,54,56,71,57,48,50,57,48,51,32,67,70,55,57,48,48, 77,68,66,32,32,32,1,22,0,0,0,7,247], "StringHexValue":"4D4549343236384739303239303320434637393 30304D444220202001160000007F7", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

16. Coin expansion feature enable

Command from user app	
{"Message":"CoinExpansionFeatureEnable", "AlternativePayout":1, "ExtendedDiagnostic":0, "ManualFillPayout":0, "FTL":0}	This command will send coin expansion feature enable command to the MDB bus. If the field value is 0 – the feature is disabled If the field value is 1 – the feature is enabled
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

17. Coin tube status

Command from user app	
{"Message":"CoinTubeStatus"}	This command will send coin tube status command to the MDB bus.
Device answer	
{"NumberValue":[0,0,55,244,0,0,0,0,0,0,0,0,0,0,0,0,0,43], "StringHexValue":"000037F4000000000000000000000000002B", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

18. Coin enable

Command from user app	
{"Message":"CoinEnable"}	This command will send all coins enable command to the MDB bus.
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

19. Coin disable

Command from user app	
{"Message":"CoinDisable"}	This command will send all coins disable command to the MDB bus.
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

20. Coin payout

Command from user app	
{"Message":"CoinPayout","CoinPosition":1,"CoinNumber":3}	This command will send coin payout command to the MDB bus. "CoinPosition" is the coin position in the coin type vector (0-based). In this example, we have sent the command to dispense 3 coins with coin type 1. Coin number must be a maximum of 15 for each command (0x0F), due to MDB value representation on 4bits.
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

21. Coin alternative payout

Command from user app	
{"Message":"CoinAlternativePayout","CoinScaledValue":7}	This command will send coin alternative payout command to the MDB bus (for Level 3+ changers only, that are supporting this option and only when the option was enabled in the initialization stage). "CoinScaledValue" represents the scaled value (considering coin changer scaling factor). In our example, with a 10 value for changer scaling factor, we have sent a value of 0.70EUR
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

22. Coin alternative payout status

Command from user app	
{"Message":"CoinAlternativePayoutStatus"}	This command will send coin alternative payout status command to the MDB bus (for Level 3+ changers only, that are supporting this option and only when the option was enabled in the initialization stage).
Device answer	
{"NumberValue":[2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,3], "StringHexValue":"02010000000000000000000000000003", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description Following the previous command to dispense 0.70, the changer has answered that it has dispensed 2 type 0 coins and 1 type in coin (2x0.10) + (1x0.50) = 0.70

23. Coin alternative payout value poll

Command from user app	
{"Message":"CoinAlternativePayoutValuePoll"}	This command will send coin alternative payout value poll command to the MDB bus (for Level 3+ changers only, that are supporting this option and only when the option was enabled in the initialization stage).
Device answer	
{"NumberValue":[7,7], "StringHexValue":"0707", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description Following the previous command to dispense 0.70, the changer has answered that it has dispensed a total value of 7 (scaled value) = 7x10 = 70 = 0.70

24. Cashless reset

Command from user app	
{"Message":"CashlessReset", "CashlessAddress":1}	This command will send cashless reset command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

25. Cashless poll

Command from user app	
{"Message":"CashlessPoll", "CashlessAddress":1}	This command will send cashless poll command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a cashless specific status/session status For example {"NumberValue":[0,0], "StringHexValue":"0000", "MessageType":"MDBMessage"} – means JUST RESET	Confirmation answer from the demo app/device, according to MDB protocol description

26. Cashless setup config data

Command from user app	
{"Message":"CashlessSetupConfigData", "CashlessAddress":1, "VMCFeatureLevel":2, "ColumnsOnDisplay":16, "RowsOnDisplay":2, "DisplayInfo":1}	This command will send cashless setup config command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a cashless setup config data response For example : {"NumberValue":[1,2,0,40,1,2,250,9,49], "StringHexValue":"010200280102FA0931", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

27. Cashless max/min prices

Command from user app	
{"Message":"CashlessMaxMinPrices", "CashlessAddress":1, "MaxPrice":65535, "MinPrice":0}	This command will send cashless max/min prices command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

28. Cashless expansion request ID

Command from user app	
{"Message":"CashlessExpansionRequestID", "CashlessAddress":1, "ManufacturerID":"ATM", "SerialNumber":"012314", "ModelNumber":"PVBT1001", "SoftwareVersion":1101}	This command will send cashless expansion request ID command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a cashless expansion ID message, for example: {"NumberValue": [9,65,84,77,100,56,56,48,51,57,53,54,102,99,49,51,80,73,67,79,86, 69,78,68,71,73,71,65,0,2,69,170], "StringHexValue":"0941544D643838303339353666633133504943 F56454E4447494741000245AA", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

29. Cashless enable

Command from user app	
{"Message":"CashlessEnable", "CashlessAddress":1}	This command will send cashless enable command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

30. Cashless disable

Command from user app	
{"Message":"CashlessDisable", "CashlessAddress":1}	This command will send cashless disable command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

31. Cashless cancel

Command from user app	
{"Message":"CashlessCancel", "CashlessAddress":1}	This command will send cashless cancel command to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or another MDB specific response for this command (check MDB spcifications for more details)	Confirmation answer from the demo app/device, according to MDB protocol description

32. Cashless revalue limit request

Command from user app	
{"Message":"CashlessRevalueLimitRequest", "CashlessAddress":1}	This command will send cashless revalue limit request to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
MDB specific response for this command (check MDB spcifications for more details) For example: {"NumberValue":[15,255,255,13], "StringHexValue":"0FFFFF0D", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

33. Cashless vend request

Command from user app	
{"Message":"CashlessVendRequest", "CashlessAddress":1, "ItemPrice":100, "ItemNumber":3}	This command will send cashless vend request to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK or a cashless specific response for this command, for example: {"NumberValue":[5,0,100,105], "StringHexValue":"05006469", "MessageType":"MDBMessage"} – means “VEND APPROVED” with a scaled value of 100 Check MDB specifications for more details	Confirmation answer from the demo app/device, according to MDB protocol description

34. Cashless vend cancel

Command from user app	
{"Message":"CashlessVendCancel", "CashlessAddress":1}	This command will send cashless vend cancel to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
Specific answer for this command is VEND DENIED (check MDB specifications for more details) {"NumberValue":[6,6], "StringHexValue":"0606", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

35. Cashless vend success

Command from user app	
{"Message":"CashlessVendSuccess", "CashlessAddress":1, "ItemNumber":3}	This command will send cashless vend success to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

36. Cashless vend failure

Command from user app	
{"Message":"CashlessVendFailure", "CashlessAddress":1}	This command will send cashless vend failure to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

37. Cashless session complete

Command from user app	
{"Message":"CashlessSessionComplete", "CashlessAddress":1}	This command will send cashless complete to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[7,7], "StringHexValue":"0707", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

38. Cashless CASH SALE reporting

Command from user app	
{"Message":"CashlessCashSale", "CashlessAddress":1,"ItemPrice":100,"ItemNumber":3}	This command will send cashless cash sale reporting to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
{"NumberValue":[0], "StringHexValue":"00", "MessageType":"MDBMessage"} – for MDB ACK {"NumberValue":[0], "StringHexValue":"FF", "MessageType":"MDBMessage"} – for MDB NAK	Confirmation answer from the demo app/device, according to MDB protocol description

39. Cashless negative vend request

Command from user app	
{"Message":"CashlessNegativeVendRequest", "CashlessAddress":1, "ItemPrice":100, "ItemNumber":3}	This command will send cashless negative vend request to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
Specific answer for this command (check MDB specifications for more details) For example VEND DENIED: {"NumberValue":[6,6], "StringHexValue":"0606", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

40. Cashless revalue request

Command from user app	
{"Message":"CashlessRevalueRequest", "CashlessAddress":1, "Amount":50}	This command will send cashless revalue request to the MDB bus “CashlessAddress” can be 1 for the cashless poll address 0x12 and 2 for cashless poll address 0x62
Device answer	
Specific answer for this command (check MDB specifications for more details) For example REVALUE APPROVED: {"NumberValue":[13,13], "StringHexValue":"0D0D", "MessageType":"MDBMessage"}	Confirmation answer from the demo app/device, according to MDB protocol description

41. MDB send raw data

Command from user app	
{"Message":"MDBRaw","Value":[55,00]}	This command will send raw data to the MDB bus. You can use this message to send any command to MDB bus, either demo app implemented commands or not implemented commands. For this command, the MDB crc is internally calculated and added to the array. You don’t need to supply MDB CRC byte for this command. Practically this command should be enough for any implementation, if you are using it to send MDB messages according to each peripheral specifications.
Device answer	
Specific answer for this peripheral which is targeted by the command (check MDB specifications for more details)	Confirmation answer from the demo app/device, according to MDB protocol description