RASPIMDBEXE

Executive/MDB bridge to Raspberry Pi

v1.0 – 07.08.2017

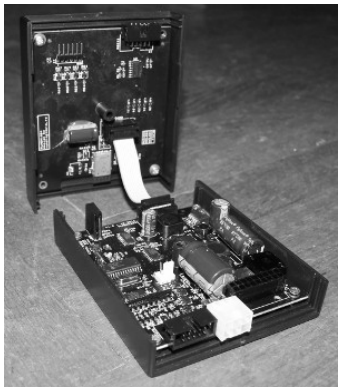


Table of Contents

I. Device overview

This device was designed to offer a quick development support for cashless systems on Executive vending machines and using MDB payment systems. It acts like a bridge between any Executive machine and MDB payment systems (bill validator, coin acceptor/changer and a cashless device). This device is working only with Level 3 or higher coin changers, because it only supports alternative payout change mode of MDB protocol.

IMPORTANT!!! - The machine must be set on Executive – price holding mode (prices are kept on the device not on the machine). It can be used also with prices on the machine, but there will be no sales informations.

It is connecting to an external device by using an RS232 or USB and offers the possibility to send some commands to the vending machine and collecting some vending informations. It can be used on any Executive price holding vending machine with a maximum of 96 selections (products).

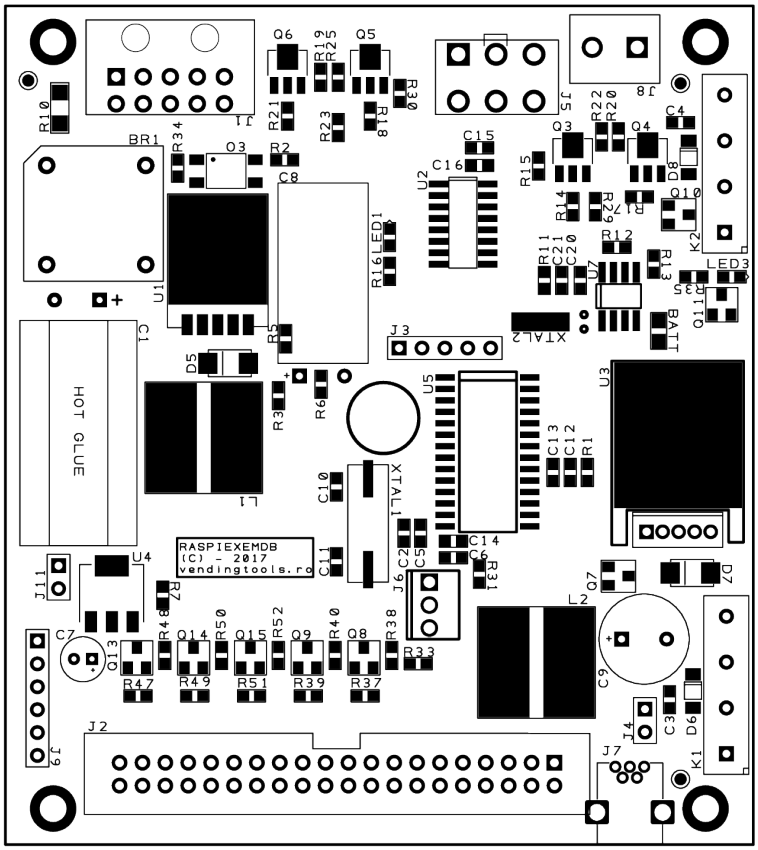
The device is self maintained and transparently manages the entire Executive, MDB and transactions flow.

The main functions available remotely by using serial port are:

- reading device memory (including sales counters, cash counters, vending settings, etc.);
- writing device memory (erasing counters, setting prices, setting vending settings, etc.);
- rising a credit to the vending machine;
- give some change to the customer;
- reading informations about the MDB cash payment systems status;
- reading informations about the current selected product on the machine’s keyboard;
- reading informations about the last transaction (vend success, vend failed);
- reading informations about the cash inserted and about the change returned to the;
- reading informations about the VMC status;
- resetting the device.

II. Hardware overview.

A. Motherboard



- J1 – Power and Executive data cable connector;

- J2 – Raspberry PI connector;

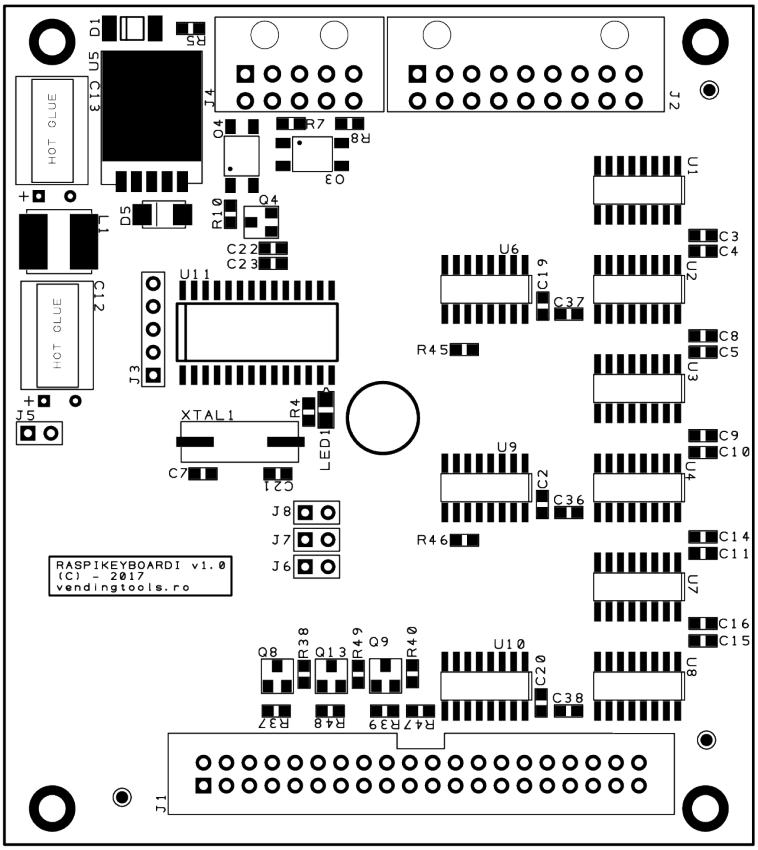
- J5 – MDB connector for payment systems;

- J8 – reserved for external 5VDC battery pack charging (not implemented in this software version);

- J7 – reserved for external 5VDC battery pack backup power (not implemented in this software version).

The device is powered by 24VAC directly from the vending machine transformer and manages to supply 24VDC/3A for MDB payment systems and 5VDC for Raspberry Pi.

B. Optional keyboard simulator board



J1 – Raspberry PI connector

J2 – Matrix keyboard connector (pin 1 to 6 = columns 1 to 6 and pin 7 to 13 = rows from 1 to 7)

C. GPIO map

Pin No.	Rpi function	RASPISLAVE	Pin No.	Rpi function	RASPISLAVE
1	3.3V	3.3V	2	5V	5V
3	GPIO2/SDA1/I2C	Not used	4	5V	5V
5	GPIO3/SCL1/I2C	Not used	6	GND	GND
7	GPIO4/GPCLK0	Not used	8	GPIO14/TXD	Serial TX
9	GND	GND	10	GPIO15/RXD	Serial RX
11	GPIO17	Not used	12	GPIO18/PCM_CLK	Not used
13	GPIO27	Not used	14	GND	GND
15	GPIO22	Not used	16	GPIO23	Not used
17	3.3V	3.3V	18	GPIO24	Not used
19	GPIO10/MOSI/SPI	OUT_DATA	20	GND	GND
21	GPIO9/MISO/SPI	OUT_LATCH	22	GPIO25	Not used
23	GPIO11/SCLK/SPI	OUT_CLOCK	24	GPIO8/CE0/SPI	Not used
25	GND	GND	26	GPIO7/CE1/SPI	Not used
27	SDA0/I2C/ID EE	Not used	28	SCL0/I2C/IDEE	Not used
29	GPIO5/GPCLK1	Not used	30	GND	GND
31	GPIO6/GPCLK2	Power good	32	GPIO12/PWM0	Not used
33	GPIO13/PWM1	Not used	34	GND	GND
35	GPIO19/PCMFS/PWM1	Not used	36	GPIO16	CTS
37	GPIO26	RTS	38	GPIO20/PCMDIN	Not used
39	GND	GND	40	GPIO21/PCMDOUT	Not used

D. Device memory map

The device's non-volatile memory contains all settings and counters needed for the device to work. The table below shows the memory map.

Address (decimal)	Size (bytes)	Description
0 → 383	384	Prices – not scaled (4 bytes for each price, from selection 1 to 96) for example a price of EUR12.50 will be represented by the following value: - B0 – 0x00 - B1 – 0x00 - B2 – 0x04 - B3 - 0xE2
384	1	Vending settings - **needs device reset command after successful write of this settings** - b0 – if set – machine is multivend, otherwise single vend - b1 – if set – machine is in price holding mode, otherwise, the prices are set on the machine - b2 – if set – machine is in force vend mode (no change without sale), otherwise it will return change even if no sale requested (can work as a change machine) - b3 – if set – machine returns no change (change inhibited) - b4 – if set – machine displays price if no credit available - b5 – b7 – reserved (not used in this version)
385 → 388	4	Maximum cash credit accepted for a transaction - **needs device reset command after successful write of this settings**
389 → 392	4	Maximum change that machine can return for a transaction - **needs device reset command after successful write of this settings**
393	1	Scaling factor (usually 10 covers most of the currencies) - **needs device reset command after successful write of this settings**
394	1	Decimal point (on the machine's display) - **needs device reset command after successful write of this settings** - if it is 0 – no decimal places - if it is 2 – one decimal - if it is 4 – two decimals - if it is 8 - three decimals
395 → 398	4	Not used in this version
399 → 403	5	Not used in this version
404 → 454	50	Not used in this version
455 → 505	50	Not used in this version
506 → 556	50	Not used in this version
557 → 607	50	Not used in this version
608 → 611	4	Not used in this version
612	1	Button status - b0 – button 1 pressed since the last ACK sent to the device - b1 – button 2 pressed since the last ACK send to the device - b2 → b7 – reserved for future use
613 → 814	207	Not used in this version
815 → 818	4	Cashless counter (accumulated cashless sales value)
819 → 822	4	Change counter (accumulated change returned value)
823 → 826	4	Bill counter (accumulated bill accepted value)
827 → 830	4	Coins counter (accumulated coins accepted value)
831 → 1023	192	Products counters (2 bytes for each product, from selection 1 to 96)

E. Keyboard registers truth table (with optional keyboard simulator board)

Col	Row	Register 2	Register 1	Register 0
1	1	0b00001000	0b00000000	0b00000000
1	2	0b00010000	0b00000000	0b00000000
1	3	0b00011000	0b00000000	0b00000000
1	4	0b00100000	0b00000000	0b00000000
1	5	0b00101000	0b00000000	0b00000000
1	6	0b00110000	0b00000000	0b00000000
1	7	0b00111000	0b00000000	0b00000000
2	1	0b00000001	0b00000000	0b00000000
2	2	0b00000010	0b00000000	0b00000000
2	3	0b00000011	0b00000000	0b00000000
2	4	0b00000100	0b00000000	0b00000000
2	5	0b00000101	0b00000000	0b00000000
2	6	0b00000110	0b00000000	0b00000000
2	7	0b00000111	0b00000000	0b00000000
3	1	0b00000000	0b00001000	0b00000000
3	2	0b00000000	0b00010000	0b00000000
3	3	0b00000000	0b00011000	0b00000000
3	4	0b00000000	0b00100000	0b00000000
3	5	0b00000000	0b00101000	0b00000000
3	6	0b00000000	0b00110000	0b00000000
3	7	0b00000000	0b00111000	0b00000000
4	1	0b00000000	0b00000001	0b00000000
4	2	0b00000000	0b00000010	0b00000000
4	3	0b00000000	0b00000011	0b00000000
4	4	0b00000000	0b00000100	0b00000000
4	5	0b00000000	0b00000101	0b00000000
4	6	0b00000000	0b00000110	0b00000000
4	7	0b00000000	0b00000111	0b00000000
5	1	0b00000000	0b00000000	0b00001000
5	2	0b00000000	0b00000000	0b00010000
5	3	0b00000000	0b00000000	0b00011000
5	4	0b00000000	0b00000000	0b00100000
5	5	0b00000000	0b00000000	0b00101000
5	6	0b00000000	0b00000000	0b00110000
5	7	0b00000000	0b00000000	0b00111000
6	1	0b00000000	0b00000000	0b00000001
6	2	0b00000000	0b00000000	0b00000010
6	3	0b00000000	0b00000000	0b00000011
6	4	0b00000000	0b00000000	0b00000100
6	5	0b00000000	0b00000000	0b00000101
6	6	0b00000000	0b00000000	0b00000110
6	7	0b00000000	0b00000000	0b00000111

III. Low level communication protocol

A. Serial port settings

The low level communication protocol can be used either on RS232 or USB and needs a hardware flow control (RTS/CTS)

Communication parameters are:

- baud rate – 115200 (could be delivered with other baud rates upon request);
- data bits – 8;
- stop bits – 1;
- parity – None.

B. Low level protocol messages

Every message consists of a message header, message data and CRC. The CRC is calculated by an XOR operation of the message header and message data bytes.

Every time a command or a response is sent, the other device should answer with ACK or NACK, accordingly.

ACK message is: 0xFC 0xFC 0xFC 0xFC 0x00

NACK message is: 0xFD 0xFD 0xFD 0xFD 0x00

As a convention for the protocol description tables, B0, B1, … Bn represents “byte number” and b0, b1, … bn represents “bit number”. Also, we will refer to this interface as “device” or as “bridge”

1. Write to memory

Using this command the data is written into the device non-volatile memory. The maximum length of data that could be written with one command is 64bytes. The device’s internal memory store the vending settings, counters, prices, etc. It is possible to modify, at any time, any of the device’s settings, reset the counters, etc. After modifying the vending configuration, the user application must send a reset command to the device. Prices and counters modification does not need a reset.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x01	- B2-B3 – beginning address to write (for example 0x02 0x58 means that the first byte will be written at address decimal 600) - B4-Bn – data bytes to be written starting with address [B2:B3]	CRC
Device response			
		ACK or NACK, according to the command execution result	

2. Read from memory

Using this command the data could be read from device's non-volatile memory. The maximum length of data that could be read with one command is 64 bytes. The device's internal memory store the vending settings, counters, prices, etc. It is highly recommended to read memory locations after write operations, to ensure that data was correctly written.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x02	- B2-B3 – beginning address to read (for example 0x02 0x58 means that the first byte will be read from address decimal 600) - B4-B5 – data length (maximum 64)	CRC
Device response			
0xFA	0x02	- B2 → Bn – data bytes read from memory	CRC

3. Device reset

Using this command the user's application can reset the device at any moment. After reset, the vending settings are reloaded and the current credit is set to 0.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x03	No parameters	0xF9
Device response			
		ACK or NACK, according to the command execution result	

4. Get device status

Using this command, at any time, the device can be interrogated about it's and it's peripherals status.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x09	No parameters	CRC
Device response			
0xFA	0x09	Note: - the response CMD is 0x09 and not 0x04!!! Status message - B2 – B16 – reserved, always 0x20 - B17 – B24 – bill validator status history (B24 contains the latest status – codes are based on the MDB bill validator answers on MDB poll command) - B25 – B32 – coin acceptor status history (B32 contains the latest status – codes are based on the MDB coin answers answers on MDB poll command) - B33 – B40 – cashless device status history (B40 contains the latest status – codes are based on the MDB cashless answers on MDB poll command) - B41 – B44 – VMC status history (0x00 – machine is up and running, 0x40, machine is out of order) - B45 – B48 – coins value counter - B49 – B52 – bills value counter - B53 – B56 – cashless value counter - B57 – B60 – change value counter - B61 – B64 – tube status (total coins value available in changer tubes) - B65 – Button pressed since last status message or since the last ACK that was sent to the device (1 = button 1, 2 = button 2, 3 = both buttons) - B66 – B69 – current cash credit - B70 – B73 – current cashless credit Most of the above values may be modified by using writemem command at the right address (see memory map table on page 6)	CRC

5. Send some credit to the machine

Using this command the user's application can send a credit to the vending machine in order to offer the possibility to select and sell a product. If the credit is higher than the price of the product selected by the customer, the difference will be ignored and cleared after the transaction, if the transaction is successful. When the customer is selecting a product, the device will return an unsolicited message containing selection number and selection price (see below – "Product selected – unsolicited message"). At the end of the transaction, the device will return an unsolicited message containing the transaction result (success or failed) according to the machine's response, based on Executive protocol.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x06	- B2 → B5 – The value (not scaled) that the user application needs to rise on the machine	CRC
Device response			
		ACK or NACK, according to the command execution result	

6. Return change to the customer

Using this command the user's application can send a command to force return change tu the customer.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x08	- B2 → B5 – The value (not scaled) that the user application needs to return to the customer	CRC
Device response			
		ACK or NACK, according to the command execution result	

7. Request current credit

This command can be used at any time to check if the machine has some credit. The "Send credit" command also modify the amount reported by this command.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0C	None	CRC
Device response			
0xFA	0x0C	- B2 → B5 – The value (not scaled) of the current credit on the machine. For example, for EUR10.00 the values will be: - B2 = 0x00 - B3 = 0x00 - B4 = 0x03 - B5 = 0xE8 - B6 → B9 – The value (not scaled) of the current cashless credit on the machine (if an MDB cashless device is connected and the cashless media is inserted).	CRC

8. Product selected – unsolicited message

This message could be received at any time, if the customer selects a product. It can be received even if there is no credit on the machine. The user's application can decide to send the corresponding credit to the device. In this situation, the machine will automatically sell the selected product. The credit must be sent in a short interval that depends on the machine (usually 8-10 seconds).

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0A	- B2 – selection number - B3 → B6 – selection price	CRC

9. Transaction result – unsolicited message

This message it is received at the end of every transaction, to inform the user's application about it.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0B	- B2 – transaction result (0x00 – transaction failed, 0x01 – transaction successful)	CRC

10. Bill accepted by the vending machine – unsolicited message

This message is received every time a bill is accepted and stacked.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0D	- B2 – B5 – the value (not scaled) of the last bill accepted and stacked by the bill validator For example, for EUR10.00 the values will be: - B2 = 0x00 - B3 = 0x00 - B4 = 0x03 - B5 = 0xE8	CRC

11. Coin accepted by the vending machine – unsolicited message

This message is received every time a coin is accepted and sorted.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0E	- B2 – B5 – the value (not scaled) of the last coin accepted and sorted by the bill validator For example, for EUR1.00 the values will be: - B2 = 0x00 - B3 = 0x00 - B4 = 0x00 - B5 = 0x64	CRC

12. Cashless in session – unsolicited message

This message is received every time the attached MDB cashless device is opening a cashless session, if the cashless credit is not 0.

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x10	- B2 – B5 – the value (not scaled) of the credit available on cashless media For example, for EUR1.00 the values will be: - B2 = 0x00 - B3 = 0x00 - B4 = 0x00 - B5 = 0x64	CRC

13. Cashless finished session – unsolicited message

This message is received every time the attached MDB cashless is closing the current session

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x10	- B2 – B5 – always 0xFFFFFFFF	CRC

14. Change requested – unsolicited message

This message is received every time a customers press the change request button or lever or every time the device is sending “Return change” command

<HEADER> (B0)	<CMD> (B1)	<PARAMETERS> (B2 to Bn)	<CRC> (Bn + 1)
0xFA	0x0F	- B2 – B5 – the value (not scaled) of the last coin accepted and sorted by the bill validator For example, for EUR1.00 the values will be: - B2 = 0x00 - B3 = 0x00 - B4 = 0x00 - B5 = 0x64	CRC

IV. High level protocol

A. Overview

The high level protocol is available by using our demo application that is connecting to the serial port and that can be accessed by the user's application through a socket. The application requires Python 3 to be installed on the user's application computer and, also, depends on PySerial 3.0.1.

The demo application requires one parameter (serial port name) and listens on port 5127 TCP.

You can use telnet or other similar application to test the daemon, sending simple commands and receiving JSON response messages.

The JSON answer is a list containing the hexadecimal values of the binary low level protocol in chapter III or an interpreted message, depending on the issued command.

B. Protocol description

1. Write to memory - METWRITEMEM

This command writes a set of bytes starting at a specified address. The data set length cannot be longer than 64 bytes. The dataset is a list of bytes, in decimal format.

Command
Example, setting second price to 1600 (EUR16.00) METWRITEMEM(4,0,0,6,64) First parameter of this function is the address to start writing, the rest of the parameters are byte values to write
Device response
{"VMCMessage":[\xFC,\xFC,\xFC,\xFC,\x00]} – ACK {"VMCMessage":[\xFD,\xFD\xFD,\xFD,\x00]} – NACK

2. Read from memory - METREADMEM

This command writes a set of bytes starting at a specified address. The data set length cannot be longer than 64 bytes. The dataset is a list of bytes, in decimal format.

Command
Example, reading second price METREADMEM(4,4) First parameter of this function is the address to start reading, the second parameter is the length (in bytes) to read
Device response
{DeviceMessage": "ReadMemory", "StartAddress": 100, "DataLegth": 3, "MemoryData": [\x00,\x00,\x00]}

3. Send credit to the machine - METSENDCREDIT(value)

This command sends a credit to the machine (not scaled value)

Command
Example, sending EUR1.60 to the machine METSENDCREDIT(160)
Device response
{"VMCMessage":[\xFC,\xFC,\xFC,\xFC,\x00]} – ACK {"VMCMessage":[\xFD,\xFD\xFD,\xFD,\x00]} – NACK

4. Return change from the machine - METSENDCHANGE(value)

This command returns change (not scaled value)

Command
Example, return EUR1.60 from changer METSENDCHANGE(160)
Device response
{"VMCMessage":[\xFC,\xFC,\xFC,\xFC,\x00]} – ACK {"VMCMessage":[\xFD,\xFD\xFD,\xFD,\x00]} – NACK

5. Get device status - METGETSTATUS

This command returns a JSON corresponding to the peripherals and counters status.

Command
METGETSTATUS
Device response
{DeviceMessage":"DeviceStatus","DeviceName":"METbrdg v1.0280", "BillStatus":[\x90,\x00,\x80,\x00,\x90,\x00,\x80,\x00], "CoinStatus":[\x00,\x00,\x00,\x00,\x00,\x00,\x0b,\x00], "CashlessStatus":[\x00,\x00,\x00,\x00,\x00,\x00,\x00,\x00], "VMCStatus":\x00,\x00,\x00,\x00], "CoinsCounter":"670", "BillsCounter":"10400", "CashlessCounter":"0", "ChangeCounter":"1210", "AvailableChange":"6700", "Button1":"NotPressed", "Button2":"NotPressed"}

6. Device reset - METRESET

This command will reset the device

Command
METRESET
Device response
{"VMCMessage":[\xFC,\xFC,\xFC,\xFC,\x00]} – ACK {"VMCMessage":[\xFD,\xFD\xFD,\xFD,\x00]} – NACK

7. Read product price - METREADPRICE(n)

Using this command the user's application can read the price of specified product. Prices are stored into the device's non-volatile memory

Command	Description
METREADPRICE(n)	- Reads the non-scaled price for the product number "n". First product is product number 1, last product is the product number 96 Below is the answer to the METREADPRICE(5) command
Device response	
{"DeviceResponse":"ReadPrice","ProductNumber":"5","ProductPrice":"150"}	

8. Write product price – METWRITEPRICE(n,value)

Using this command the user's application can write the price of specified product. Prices are stored into the device's non-volatile memory

Command	Description
METWRITEPRICE(n,value)	- Writes the non-scaled price for the product number "n". First product is product number 1, last product is the product number 96. For example METWRITEPRICE(5,150) sets the price of the 5th selection to EUR 1.50
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

9. Read maximum credit setting - METREADMAXCREDIT

Using this command the user's application can read the maximum credit set on the device. While a transaction, if the cash inserted by the customer reaches this limit, the payment systems are inhibited. It Is a safety method to avoid returning too much change.

Command	Description
METREADMAXCREDIT	- Reads the non-scaled MAXIMUM CREDIT settings for transactions. Below is the an example answer to the METREADMAXCREDIT command (maximum credit is set to EUR 10.00)
Device response	
{"DeviceResponse":"ReadMaxCredit","MaxCreditValue":"1000"}	

10. Write maximum credit setting - METWRITEMAXCREDIT(value)

Using this command the user's application can write the maximum credit set on the device. While a transaction, if the cash inserted by the customer reaches this limit, the payment systems are inhibited. It Is a safety method to avoid returning too much change.

Command	Description
METWRITEMAXCREDIT(value) **Requires METRESET after issuing this command, since the value is loaded only once, at start-up**	- Writes the non-scaled MAXIMUM CREDIT accepted for a transaction. For example set max credit to EUR 5:.00 METWRITEMAXCREDIT(500)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

11. Read maximum change setting - METREADMAXCHANGE

Using this command the user's application can read the maximum change permitted after a transaction. If the credit remaining after a transaction is higher than this value, them the machine will not return change. It is used also as a safety method to avoid returning too much change.

Command	Description
METREADMAXCHANGE	- Reads the non-scaled MAXIMUM CHANGE settings for transactions. Below is the an example answer to the METREADMAXCHANGE command (maximum credit is set to EUR 9.00)
Device response	
{"DeviceResponse":"ReadMaxChange","MaxChangeValue":"900"}	

12. Write maximum change setting - METWRITEMAXCHANGE(value)

Using this command the user's application can write the maximum credit set on the device. While a transaction, if the cash inserted by the customer reaches this limit, the payment systems are inhibited. It Is a safety method to avoid returning too much change.

Command	Description
METWRITEMAXCHANGE(value) **Requires METRESET after issuing this command, since the value is loaded only once, at start-up**	- Writes the non-scaled MAXIMUM CHANGE accepted for a transaction. For example set max change to EUR 5.00: METWRITEMAXCHANGE(500)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

13. Read scaling factor - METREADSCALINGFACTOR

Using this command the user's application can read the scaling factor on the device. Usually is is a good practice to keep the scaling factor sync-ed with the machine's scaling factor setting. Some machines are ignoring this and work on their scaling factor. In that case, if they are not matching, this will drive you to display error on values.

Command	Description
METREADSCALINGFACTOR	- Reads the SCALING FACTOR settings credit and price display. Below is the an example answer to the METREADSCALINGFACTOR command (where scaling factor is 10)
Device response	
{"DeviceResponse":"ReadScalingFactor","ScalingFactorValue":"10"}	

14. Write scaling factor – METWRITESCALINGFACTOR(value)

Using this command the user's application can write the scaling factor set on the device. Usually is is a good practice to keep the scaling factor sync-ed with the machine's scaling factor setting. Some machines are ignoring this and work on their scaling factor. In that case, if they are not matching, this will drive you to display error on values.

Command	Description
METWRITESCALINGFACTOR(value) **Requires METRESET after issuing this command, since the value is loaded only once, at start-up**	- Writes the SCALING FACTOR setting for credit/price display. For example set scaling factor to 10: METWRITESCALINGFACTOR(10)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

15. Read decimal point - METREADDECIMALPOINT

Using this command the user's application can read the decimal point on the device. Usually is is a good practice to keep the decimal point sync-ed with the machine's decimal point setting. Some machines are ignoring this and work on their decimal point setting. In that case, if they are not matching, this will drive you to display error on values.

Command	Description
METREADDECIMALPOINT	- Reads the DECIMAL POINT settings for credit/price display. Below is an example answer to the METREADDECIMALPOINT command (where decimal point is 2, meaning 2 decimals after the decimal point):
Device response	
{"DeviceResponse":"ReadDecimalPoint","DecimalsNumber":"2"}	

16. Write decimal point – METWRITEDECIMALPOINT(value)

Using this command the user's application can write the decimal point on the device. Usually is is a good practice to keep the decimal point sync-ed with the machine's decimal point setting. Some machines are ignoring this and work on their decimal point setting. In that case, if they are not matching, this will drive you to display error on values.

Command	Description
METWRITEDECIMALPOINT(value) **Requires METRESET after issuing this command, since the value is loaded only once, at start-up**	- Writes the DECIMAL POINT. For example set decimal point to 2: METWRITESCALINGFACTOR(2)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

17. Read coins counter - METREADCOINSCOUNTER

Using this command the user's application can read the non-volatile counter representing the total value of the cashed coins.

Command	Description
METREADCOINSCOUNTER	Reads the COINS COUNTER Below is an example where the device shows a value or EUR 6.70 for the cashed coins.
Device response	
{"DeviceResponse":"ReadCoinsCounterValue","CoinsCounterValue":"670"}	

18. Write coins counter – METWRITECOINSCOUNTER(value)

Using this command the user's application can write the non-volatile counter representing the total value of the cashed coins. It can be used to set this value to 0 when the vending machine operator is performing a cash-collect operation.

Command	Description
METWRITECOINSCOUNTER(value)	- Writes the coins counter. For example resetting the counter on cash-collect: METWRITECOINSCOUNTER(0)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

19. Read bills counter - METREADBILLSCOUNTER

Using this command the user's application can read the non-volatile counter representing the total value of the cashed bills.

Command	Description
METREADBILLSCOUNTER	Reads the BILLS COUNTER Below is an example where the device shows a value or EUR 104.00 for the cashed bills.
Device response	
{"DeviceResponse":"ReadBillsCounterValue","BillsCounterValue":"10400"}	

20. Write bills counter – METWRITEBILLSCOUNTER(value)

Using this command the user's application can write the non-volatile counter representing the total value of the cashed bills. It can be used to set this value to 0 when the vending machine operator is performing a cash-collect operation.

Command	Description
METWRITEBILLSCOUNTER(value)	- Writes the bills counter. For example resetting the counter on cash-collect: METWRITEBILLSCOUNTER(0)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

21. Read change counter - METREADCHANGECOUNTER

Using this command the user's application can read the non-volatile counter representing the total value of the returned change to the customers.

Command	Description
METREADCHANGECOUNTER	Reads the CHANGE COUNTER Below is an example where the device shows a value or EUR 12.10 for the returned change.
Device response	
{"DeviceResponse":"ReadChangeCounterValue","ChangeCounterValue":"1210"}	

22. Write change counter – METWRITECHANGECOUNTER(value)

Using this command the user's application can write the non-volatile counter representing the total value of the returned change. It can be used to set this value to 0 when the vending machine operator is performing a cash-collect operation.

Command	Description
METWRITECHANGECOUNTER(value)	- Writes the change counter. For example resetting the counter on cash-collect: METWRITECHANGECOUNTER(0)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

23. Read cashless counter - METREADCASHLESSCOUNTER

Using this command the user's application can read the non-volatile counter representing the total value of the cashless transactions.

Command	Description
METREADCASHLESSECOUNTER	Reads the CASHLESS COUNTER Below is an example where the device shows a value or EUR 12.50 for cashless transactions.
Device response	
{"DeviceResponse":"ReadCashlessCounterValue","CashlessCounterValue":"1250"}	

24. Write cashless counter – METWRITECASHLESSCOUNTER(value)

Using this command the user's application can write the non-volatile counter representing the total value of the cashless transactions. It can be used to set this value to 0 when the vending machine operator is performing a cash-collect operation.

Command	Description
METWRITECASHLESSCOUNTER(value)	- Writes the cashlesscounter. For example resetting the counter on cash-collect: METWRITECASHLESSCOUNTER(0)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

25. Read product counter - METREADPRODUCTCOUNTER(n)

Using this command the user's application can read the non-volatile counter representing the total (pieces) sales for a product.

Command	Description
METREADPRODUCTCOUNTER(n)	Reads the product Below is an example where the device shows a number of 16 poducts sold for selection number 1 as an answer to METREADPRODUCTCOUNTER(1)
Device response	
{"DeviceResponse":"ReadProductCounter","ProductNumber":"1","ProductSalesNumber":"16"}	

26. Write product counter – METWRITEPRODUCTCOUNTER(n,value)

Using this command the user's application can write the non-volatile counter representing the total (pieces) sales for a product. It can be used to set this value to 0 when the vending machine operator is performing a refill operation.

Command	Description
METWRITEPRODUCTCOUNTER(n,value)	- Writes the product counter . For example resetting the counter for product 1 on refill: METWRITEPRODUCTCOUNTER(1,0) Product number can take values between 1 and 96 and number of sales can take values between 0 and 65535 (2 bytes)
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

27. Read vending settings - METREADVENDINGSETTINGS

Using this command the user's application can read the device vending settings

Command	Description
METREADVENDINGSETTINGS	Reads the device vending settings.
Device response	
{"DeviceResponse":"ReadVendingSettings","VendingSettingsValue":"7"} The value is representing the byte at address 384 in memory map table (page 6)	

28. Write vending settings – METWRITEVENDINGSETTINGS(value)

Using this command the user's application can write the non-volatile vending settings, according to the byte at address 384 in memory map (page 6).

Command	Description
METWRITEVENDINGSETTINGS(value) **Requires METRESET after issuing this command, since the value is loaded only once, at start-up**	Writes the vending settings in the device's memory
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

29. Read current credit - METREADCURRENTCREDIT

Using this command the user's application can read the vending machine current credit

Command	Description
METREADCURRENTCREDIT	Reads the vending machine current credit. Below is an answer example when the machine has a credit of EUR5.20
Device response	
{"DeviceResponse":"ReadCurrentCredit","CurrentCreditValue":"520"}	

30. Write buttons value – METWRITEBUTTONSVALUE(value)

Using this command, after reading the buttons status with METGETSTATUS, the value could be erased to wait the next press. The command will write the byte on address 612 on memory map (page 6) and hase the same structure (bit 0 → button 1, bit 1 → button 2)

Command	Description
METWRITEBUTTONSvalue)	Writes a value on the buttons location, usually used to clear the buttons status after reading with METGETSTATUS.
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

30. Send a keypress to the virtual keyboard simulator – METKEYPRESS(column,row,delay) – only with additional Raspberry Pi keyboard simulator board.

Using this command, the Raspberry Pi board will manipulate GPIO in order to send the appropriate impulse to the machine's matrix keyboard. Raspberry Pi will manipulate 3 of the GPIOs to select simulator board MUX channel.

Command	Description
METKEYPRESS(column,row,delay)	- column is the selected column that will be connected to the row - row is the selected column that will be connected to the column - delay is the time (milliseconds) that column and row will be connected together. For GPIO line manipulation, please check "Keyboard registers truth table" and Raspberry PI GPIO map.
Device response	
{"DeviceResponse":"Acknowledge","MessageID":"101"} or {"DeviceResponse":"Negative Acknowledge","MessageID":"102"}	

C. Unsolicited messages

1. Vend request

This message is send by the device, every time the customer is selecting a product and there is a corresponding credit that covers the product's price.

Message
{"VMCMessage": "VendRequest", "ProductID": 7, "ProductPrice": 130, "MessageID": 103}
Description
This message contains informations about the last selected product on the vending machine's keyboard. - "ProductID" is the number of the latest product number selected by the customer (key number/selection number); - "ProductPrice" is the price of the latest product selected by the customer; - "MessageID" is the message ID number, always 103 for this message

2. Vend result

This message is send by the device, every time the VMC is finishing a transaction (product dispensed or product failed to dispense).

Message
{"VMCMessage": "VendResult", "ResultCode": "VendSuccess", "MessageID": 104}
Description
This message contains informations about the result of the last transaction. - "ResultCode" is the human readable transaction result and can be "VendSuccess" or "VendFailed" - "MessageID" is the message ID number, and it can be 104 for "VendSuccess" and 105 for "VendFailed"

3. Bill accepted

This message is send by the device, every time a bill is inserted into the bill validator and stacked.

Message
{DeviceMessage": "BillAccepted","BillAcceptedValue": 500}
Description
"BillAcceptedValue" is the value of the last accepted bill (in this example, bill of EUR5.00).

4. Coin accepted

This message is send by the device, every time a coin is inserted into the coin acceptor and sorted to a tube or to the cash-box.

Message
{DeviceMessage": "CoinAccepted","CoinAcceptedValue": 50}
Description
"CoinAcceptedValue" is the value of the last accepted coin (in this example, coin of EUR0.50).

5. Change requested

This message is send by the device, every time a change request is sent to the coin changer.

Message
{DeviceMessage": "ChangeRequested","ChangeRequestedValue": 150}
Description
“ChangeRequestedValue” is the value of the last change command sent to the coin changer (in this example EUR1.50).

6. Cashless finish session

This message is send by the device, every time the cashless device is finishing the session (cashless support removed).

Message
{"DeviceMessage": "CashlessFinishSession","CashlessCreditValue": 0}
Description

7. Cashless start session

This message is send by the device, every time the cashless device starting a new session (cashless support presented to the cashless device).

Message
{"DeviceMessage": "CashlessStartSession","CashlessCreditValue": 1250}
Description
- “CashlessCreditValue” is the value of the cashless credit available on the customer’s support.

8. Button pressed

This message is send by the device, every time a button is pressed.

Message
{"DeviceMessage": "ButtonPressed","ButtonNumber": 1}
Description
- “ButtonNumber” is the number of the pressed button (in this example, button #1)

Notes: