Raspberry PI vending solution (RASPIVEND) v.28.11.2016 Quick Reference

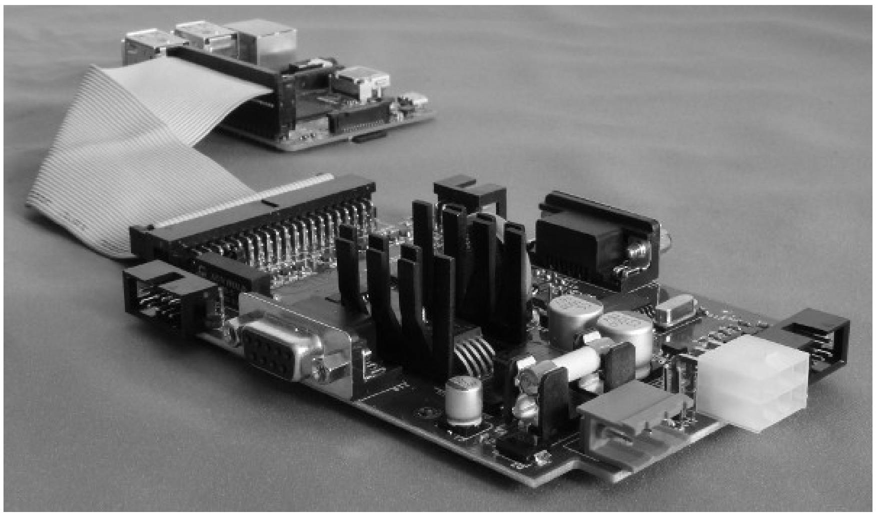


Table of Contents

I. General informations

1. Terms

- **RASPIVEND** = Raspberry PI vending board (shield).
- **RASPIVEND DAEMON** - xinetd based RASPIVEND management daemon.
- **MDB PERIPHERALS** = payment systems connected on the MDB bus.
- **CCTALK PERIPHERALS** = payment systems and peripherals, connectod to ccTalk bus.
- **HOST APPLICATION** = the application executed by the xinetd superserver installed on the Raspberry PI
- **CLIENT APLICATION** = the client application that will connect to the socket of the HOST APPLICATION
- **LOW LEVEL APPLICATION** = the application that can directly communicate using Raspberry PI serial port (/dev/ttyAMA0) and also can handle it's GPIO pins.
- **ACK** = acknowledge
- **NACK** = not-acknowledge

RASPIVEND board and RASPIVEND daemon, toghether with VTLCOMBUS keyboard simulator module, can be used to upgrade any Necta hot and universal machines and alos any Wurlitzer universal spring machine.

2. Working modes

The RASPIVEND can be used to communicate with peripherals using two methods:
a. A low level communication method that can offer access to all peripherals by the Raspberry PI serial port (/dev/ttyAMA0) and using 3 of it's GPIO to handle the multiplexers to select the proper communication channel.
b. A high level communication method that simplifies the user interface development, offering a language independent support.

A. Low level mode

In low level mode, there are simple commands to manage the MDB devices. The built-in firmware will handle MDB commands and it is the ideal mode where the developers don't need to learn any MDB command and response. Also there is no need to calculate the MDB checksum since this is automatically calculated by the RASPIVEND and correctly sent to the MDB peripheral. When the RASPIVEND will receive a low level command, it automatically turn low level mode ON and begin to continuously poll the MDB PERIPHERALS.
In this mode, there is no limit accessing any other peripheral and the LOW LEVEL APPLICATION (created by the user) has the entire responsibility of multiplexer controls pin manipulation by using the appropriate GPIO. The only channel that is automatically selected is the MDB channel, which is selected every time a MDB low level command is sent to the board (see page 9 for pin usage).
A proprietary simple message structure is available to communicate with bill validators, coin acceptors/changers and cashless devices. The general message format is detailed in table 1.

B. High level mode

This mode is using a xinetd driven application that can use a socket to communicate.

For the moment, there are some limitations, depending on the level mode, but we are constantly work to add more and more functions on the high level mode too, because this mode can be used with less or no complications, even with a browser based application.

Below you can find a table with a level modes comparison:

Function/Level mode	Low level	High level
MDB coin acceptor/changer	yes	yes
MDB bill validator	yes	yes
MDB cashless #1	yes	yes
MDB cashless #2	yes	yes
ccTalk	yes	Only with Python 3
RS232 #1	yes	yes
RS232 #2	yes	yes
GPRS communication port	yes	yes
VTLCOMBUS Necta keyboard	Only with xinetd	Only with xinetd
VTLCOMBUS Wurlitzer keyboard	Only with xinetd	Only with xinetd
RTC set/get	Only with xinetd	Only with xinetd
Raspberry PI GPIO	yes	future development
POWER GOOD signal for autoshutdown	future development	future development
Direct peripheral selection and communication	yes	yes

Table 1: Level modes comparison

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
always 0xFE	1 byte	1 byte	variable length, depending on subcommand	1 byte

Table 2: Low level message format

- **<HEADER>** – Is always a byte with a value of 0xFE.
- **<CMD>** - Defines the command group sent to the RASPIVend.
- **<SUBCMD>** - Defines the specific subcommand of the command group.
- **<PARAMETERS>** - Defines the command parameters that will be send to the MDB peripheral (for example, the maximum credit that the INTERFACE should accept or the change it should return from changer). Some commands are not requiring parameters. Also, the variable length depends on subcommand.
- **<CRC>** - Defines the message checksum. The <CRC> is calculated as an XOR of all message bytes, including the <HEADER>. For example, the command to enabled the bill validator is: 0xFE 0x42 0x02 and the CRC for this command is 0xBE. Transparent mode can be used for any application.

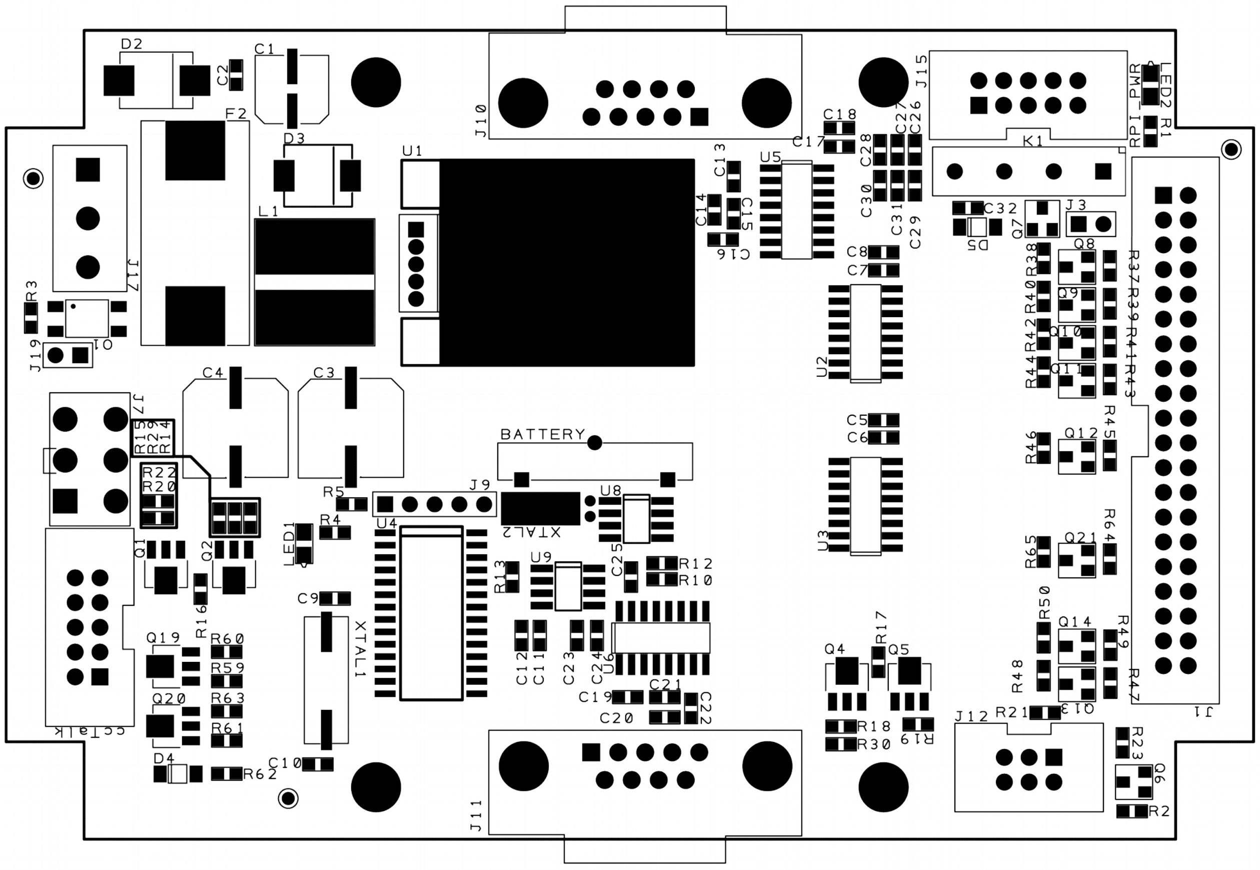
2. Communication parameters

The communication settings should meet the following specifications:
a. For the peripherals (excepting the MDB bus), there is no restriction regarding the serial port settings you need.
b. For then MDB communication parameters:

Parameter	Value
baud	57600
data bits	8
parity	NONE
hardware flow	YES (RTS/CTS)
software flow	NO

Table 3: MDB communication parameters

II. Hardware overview


Picture 1: Board overview

1. Power supply requirements

The RASPIVEND can be powered with stabilized 24VDC or 12VDC, depending on your MDB PERIPHERALS and CCTALK Peripherals. You must use a stabilized DC power supply with at least 2A output. It is necessary to follow the correct polarity. In the eventuality of an accidental polarity reversal, the entire board, the MDB PERIPHERALS and the CCTALK PERIPHERALS are protected, but will not work. The board also supplies the 5V/2A for Raspberry PI (or compatible). The system eliminates the separate 5V microUSB power supply for Raspberry PI. You will only need one power supply for the entire system.

2. Connector description

- **<J17> –** POWER connector for the RASPIVEND and MDB PERIPHERALS. Use only stabilized power supplies, with a voltage rating according to your MDB PERIPHERALS. Also, be careful at the current rating, since this may vary from one MDB peripheral to another. Use your MDB peripheral manual to identify the power needs.
- **<J10>** - RS232 connector. General purpose RS232 serial port, with no hardware flow wires.
- <J11> - RS232 connector. General purpose RS232 serial port, with no hardware flow

wires
- **<J7>** - Used to connect the MDB PERIPHERALS.
- <**ccTalk>** - Used to connect he CCTALK PERIPHERALS.
- **<J12>** - VTLCOMBUS. This is a proprietary protocol that can be used to expand the board with any needed boards (I/O ans sensr boards, etc.). For the moment there is one single board available for this device and it's function is to simulate keyboard press for Necta hot and spring machines and for Wurlitzer universal spring machines.
- **<J15>** - Communication port. For the moment there is one device available for this port and it is a GPRS communication module based on SIMCOM M2M block.
- **<J1> -** 40 pins Raspberry PI (or compatible) single board computer connector. This connector provides access to POWER, GPIO and serial port. This connector, also supplies the power for Raspberry PI. Used pins and functions can be find in the table below. Pins marked by green background are available for user applications and alos, the power and GND pins. Pins marked with red background are reserved for RASPIVEND.

Pin No.	Rpi function	RASPIVEND	Pin No.	Rpi function	RASPIVEND
1	3.3V	3.3V	2	5V	5V
3	GPIO2/SDA1/I2C	MODEM DCD	4	5V	5V
5	GPIO3/SCL1/I2C	MODEM RTS	6	GND	GND
7	GPIO4/GPCLK0	MODEM PWR	8	GPIO14/TXD	Serial TX
9	GND	GND	10	GPIO15/RXD	Serial RX
11	GPIO17	MUX A	12	GPIO18/PCM_CLK	Modem stat
13	GPIO27	MUX B	14	GND	GND
15	GPIO22	MUX C	16	GPIO23	Not used
17	3.3V	3.3V	18	GPIO24	Not used
19	GPIO10/MOSI/SPI	Not used	20	GND	GND
21	GPIO9/MISO/SPI	Not used	22	GPIO25	Not used
23	GPIO11/SCLK/SPI	Not used	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	MDB CTS
37	GPIO26	MDB RTS	38	GPIO20/PCMDIN	Not used
39	GND	GND	40	GPIO21/PCMDOUT	Not used

You do not need to perform any settings on the RASPIVEND, neither hardware or software.

III. Low level mode

To use this mode, your application must handle the following:
a. GPIO pins of the Raspberry PI, needed to switch the multiplexers to the correct communication channel (to select the peripheral).
b. Serial port of the Raspberry PI (/dev/ttyAMA0).
c. for MDB handling you need, also, to manipulate MDB RTS and MDB CTS pins.
The MDB communication is handled by setting MDB RTS pin to lo and waiting for CTS PIN to be set as lo by the board. Only messages sent by reading lo on MDB CTS will be correctly received by the device. Please be sure that will keep MDB RTS high when you do not need to communicate to MDB bus.

1. Multiplexer GPIO truth table

GPIO22	**GPIO27**	**GPIO17**	**Selected peripheral (the peripheral currently connected to RasPI)**
0	0	0	MDB
0	0	1	RS232 #1 (J10)
0	1	0	RS232 #2 (J11)
0	1	1	VTLCOMBUS
1	0	0	ccTalk
1	0	1	Modem

2. MDB bill validator initialization

<HEADER>	**<CMD>**	**<SUBCMD>**	**<PARAMETERS>**	**<CRC>**
0xFE	0x42	0x01	[none]	0xBD
RASPIVEND answer				
0xFE	0x42	0x01	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will execute the initialization procedure for the MDB bill validator connected on the MDB port.

3. MDB bill enable

<HEADER>	**<CMD>**	**<SUBCMD>**	**<PARAMETERS>**	**<CRC>**
0xFE	0x42	0x02	[none]	0xBE
RASPIVEND answer				
0xFE	0x42	0x02	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will enable the bill validator that will start accepting all the banknotes it can recognize.

4. MDB bill disable

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x42	0x03	[none]	0xBF
RASPIVEND answer				
0xFE	0x42	0x03	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will disable the bill validator that will no longer accept any banknote.

5. MDB bill read setup vector

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x42	0x04	[none]	0xB8
RASPIVEND answer				
0xFE	0x42	0x04	- <BILL SETUP> - 27 bytes - <BILL EXPANSION IDENTIFICATION> - 29 bytes	CRC

This command will return the settings vector for the MDB bill validator. Those vectors are read on the initialization phase. There are two vectors available and the contained data are detailed in the MDB documentation. This command is optional and is used only if you need to handle some lower informations (bill validator MDB level, software version, ISO country code, etc.).

6. Coin acceptor initialization

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x43	0x01	[none]	0xBC
RASPIVEND answer				
0xFE	0x43	0x01	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will execute the initialization procedure for coin acceptor/changer connected on the MDB port.

7. Coin acceptor enable

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x43	0x02	[none]	0xBF
RASPIVEND answer				
0xFE	0x43	0x02	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will activate the coin acceptor/changer. All recognized coins/tokens will be accepted and deposited.

8. Coin acceptor disable

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x43	0x03	[none]	0xBE
RASPIVEND answer				
0xFE	0x43	0x03	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will deactivate the coin acceptor/changer.

9. Coin acceptor read setup vectors

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x43	0x04	[none]	0xB9
RASPIVEND answer				
0xFE	0x43	0x04	- <COIN SETUP> - 23 bytes - <COIN EXPANSION IDENTIFICATION> - 33 bytes	CRC

This command will return the settings vector for the MDB coin acceptor/changer. Those vectors are read on the initialization phase. There are two vectors available and the contained data are detailed in the MDB documentation. This command is optional and is used only if you need to handle some lower informations (coin acceptor/changer MDB level, software version, ISO country code, etc.).

10. Set maximum credit

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x52	0x01	<MAXIMUM CREDIT> - 4 bytes – MSB Example: 0xFE 0x52 0x01 0x00 0x00 0x02 0x58 0xF7 – this will set the maximum credit to 600 units. In case of EUR or USD, this means 600 cents or 6.00EUR/6.00USD	CRC
RASPIVEND answer				
0xFE	0x52	0x01	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will set the maximum acceptable credit for bills. Any bill exceeding this value will be rejected. For coins, you should disable the MDB coin acceptor after reaching the maximum credit value.

11. Reset the current credit

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x52	0x02	[none]	0xAE
RASPIVEND answer				
0xFE	0x52	0x02	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

The RASPIVEND has an internal cash counter which is incremented for each bill or coin deposited by the MDB PERIPHERALS. This counter can be read by using a POLL command, detailed on “12. Poll credit and devices status”. For simplicity reasons, this counter can be reseted by this command. You can use this command after each transaction, or anytime you need.

12. Return change

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x52	0x03	<CHANGE TO RETURN> - 4 bytes – MSB Example: 0xFE 0x52 0x03 0x00 0x00 0x01 0x5E 0xF0 – this command will return 350 change, which means, for EUR and USD, 350cents or 3.50EUR/3.50USD	CRC
RASPIVEND answer				
0xFE	0x52	0x03	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will start returning change if the RASPIVEND has a changer connected on the MDB port.

13. Set current cash credit

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x52	0x04	<SET CURRENT CASH CREDIT> - 4 bytes – MSB Example: 0xFE 0x52 0x03 0x00 0x00 0x01 0x5E 0xF0 – this command will set current credit to 350, which means, for EUR and USD, 350cents or 3.50EUR/3.50USD – This is used to adjust credit in multivend mode and before the cashless revalue command to set only remaining credit for revalue.	CRC
RASPIVEND answer				
0xFE	0x52	0x04	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will sest the current credit on the RASPIVEND. It is indicated to use this command after every cash transaction finished with a successful vend.

14. Poll credit and devices status

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x50	0x01	[none]	0xAF
RASPIVEND answer				
0xFE	0x50	0x01	- <CURRENT CREDIT CASH> - 4 bytes MSB (for example, 0x00 0x00 0x04 0xE2 representing 1250 cents or 12.50EUR/12.50USD) - <CURRENT CREDIT CASHLESS> - 4 bytes MSB (for example, 0x00 0x00 0x04 0xE2 representing 1250 cents or 12.50EUR/12.50USD) - <BILL validator status> - 4 byte, according to MDB bill validators status (see Table 3) – this register contains last 4 bill validator status. If the register has the value 0xFFFFFFFF, then the bill validator is not initialized. The rightmost byte of this register is the last status. - <COIN acceptor/changer status> - 4 bytes, according to MDB coin acceptors/changers status (see Table 4) - <CASHLESS #1 status> - 4 bytes, according to the MDB cashless status (see Table 5) - <CASHLESS #2 status> - 4 bytes, according to the MDB cashless status (see Table 5) - <CASHLESS MEDIA ID> - 4 bytes, card/tag serial number	CRC

This command must be run periodically, at least one per second, to interrogate the payment systems status and to take all needed decisions.

Value	Description
0x00	Idle
0x01	Defective Motor - One of the motors has failed to perform its expected assignment.
0x02	Sensor Problem - One of the sensors has failed to provide its response.
0x03	Validator Busy - The validator is busy and can not answer a detailed command right now.
0x04	ROM Checksum Error - The validators internal checksum does not match the calculated checksum.
0x05	Validator Jammed - A bill(s) has jammed in the acceptance path.
0x06	Validator was reset - The validator has been reset since the last POLL.
0x07	Bill Removed - A bill in the escrow position has been removed by an unknown means. A BILL RETURNED message should also be sent.
0x08	Cash Box out of position - The validator has detected the cash box to be open or removed.
0x09	Validator Disabled - The validator has been disabled, by the VMC or because of internal conditions
0x0A	Invalid Escrow request - An ESCROW command was requested for a bill not in the escrow position.
0x0B	Bill Rejected - A bill was detected, but rejected because it could not be identified.
0x0C	Possible Credited Bill Removal – There has been an attempt to remove a credited (stacked) bill.
0xFF	Not known or not initialized status

Table 2: Bill validators status codes

Value	Description
0x00	Idle
0x01	Escrow request - An escrow lever activation has been detected.
0x02	Changer Payout Busy - The changer is busy activating payout devices.
0x03	No Credit - A coin was validated but did not get to the place in the system when credit is given.
0x04	Defective Tube Sensor - The changer has detected one of the tube sensors behaving abnormally.
0x05	Double Arrival - Two coins were detected too close together to validate either one.
0x06	Acceptor Unplugged - The changer has detected that the acceptor has been removed.
0x07	Tube Jam - A tube payout attempt has resulted in jammed condition.
0x08	ROM checksum error - The changers internal checksum does not match the calculated checksum.
0x09	Coin Routing Error - A coin has been validated, but did not follow the intended routing.
0x0A	Changer Busy - The changer is busy and can not answer a detailed command right now.
0x0B	Changer was Reset - The changer has detected an Reset condition and has returned to its power-on idle condition.
0x0C	Coin Jam - A coin(s) has jammed in the acceptance path.
0x0D	Possible Credited Coin Removal – There has been an attempt to remove a credited coin.
0xFF	Not known or not initialized status

Table 3: Coin acceptors/changers status codes

Value	Description
0x00	Idle
0x01	Not used in this configuration
0x02	Display request – not used in this configuration
0x03	Begin cashless session
0x04	Cancel request from cashless to VMC
0x05	Vend approved.
0x06	Vend denied.
0x07	End cashless session.
0x08	Cancel
0x09	Not used in this configuration.
0x0A	Not used in this configuration.
0x0B	Command out of sequence.
0x0C	Not used in this configuration.
0x0D	Revalue approved.
0x0E	Revalue denied
0x0F	Not used in this configuration.

Table 4: Cashless status codes

15. Reset all devices' status

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x50	0x02	[none]	0xAC
RASPIVEND answer				
0xFE	0x50	0x02	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will set to 0x00 all devices' status. It is used to clear status and let the RASPIVEND to update it in accordance with the new devices' status. It will not reset credits

16. Cashless INIT

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x01	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x01	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will init the cashless device with the corresponding number

17. Cashless ENABLE

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x02	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x02	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will enable the cashless device with the corresponding number

18. Cashless DISABLE

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x03	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x03	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will disable the cashless device with the corresponding number

19. Cashless VEND CANCEL

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x04	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x04	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will send “CANCEL CURRENT SESSION” command to the cashless device.

20. Cashless VEND REQUEST

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x05	<CASHLESS NUMBER> 1 byte (0x01 = cashless #1, 0x02 = cashless #2) <VEND VALUE> - 4 bytes - the value of the selected product to sell - MSB (for example, 0x00 0x00 0x04 0xE2 representing 1250 cents or 12.50EUR/12.50USD)	CRC
RASPIVEND answer				
0xFE	0x53	0x05	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will send ask the permission to sell a product using cashless. This command requires to use POLL command to obtain the answer from the cashless device (according to Table 5).

21. Cashless VEND SUCCESS

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x06	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x06	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command must be sent after a successful vend using cashless. Following this command, the RASPIVEND will substract the product value from the cashless credit, and the cashless device will be instructed to substract th same amount from the customer's credit.

22. Cashless VEND FAIL

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x07	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x07	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command must be sent after a failed sale situation (due to VMC error). The cashless will be instructed to refund the product price to the customer's account.

23. Cashless REVALUE

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x08	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x08	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will transform the remaining cash credit into cashless credit and will refund this amount on customer's account.

24. Cashless READ VECTORS

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x09	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x09	<CASHLESS SETUP> - 8 bytes <CASHLES EXPANSION INFORMATION> - 30 bytes	CRC

This command will return the low level settings of the cashless device, according to MDB protocol.

25. Cashless END SESSION

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x53	0x0A	<CASHLESS NUMBER> (0x01 = cashless #1, 0x02 = cashless #2)	CRC
RASPIVEND answer				
0xFE	0x53	0x0A	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will return turn the session off and will force the cashless device to return to idle state.

26. RTC set

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x50	0x03	<DATE_TIME> - 7 bytes - seconds (00 -> 59) - minutes (00 -> 59) - hour (00 -> 23) - Day of week (1 -> 7) - Day of month (1 -> 31) - Month (1 -> 12) - Year (00 ->99)	CRC
RASPIVEND answer				
0xFE	0x50	0x03	0xFB – command execution failed 0xFC – command successfully executed 0xFD – command CRC error	CRC

This command will set the date and hour into the RTC.

27. RTC get

<HEADER>	<CMD>	<SUBCMD>	<PARAMETERS>	<CRC>
0xFE	0x50	0x04	[none]	CRC
RASPIVEND answer				
0xFE	0x50	0x04	<DATE_TIME> - 7 bytes - seconds (00 -> 59) - minutes (00 -> 59) - hour (00 -> 23) - Day of week (1 -> 7) - Day of month (1 -> 31) - Month (1 -> 12) - Year (00 ->99)	CRC

This command will read and return the date and hour from the RTC.

IV. High level mode with xinetd

Using this mode, the development becomes much easier. The communication with the board and the peripherals is managed by a small application configured to be handled by the xinetd super-server. The application will be delivered with the RASPIVEND.
Commands are not case sensitive. We have used capitalization to facilitate reading.
It is a good idea for your application to retry sending the command few times if you get an “failed” answer. This answer can be returned in the event of Raspberry PI to RASPIVEND board communication failure. Also it can be returned if you try to address a not connected device.

1. Configuring xinetd

Proceed to xinetd installation, according to your distribution specifications.

2. MDBBillInit

GUI command	
Command	**Parameters/Comments**
MDBBillInit	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillInit":"success"}
or
{"MDBBillInit":"failed"}
	This command will perform all initialization tasks for the attached MDB bill validator. If something goes wrong or the MDB bill validator is not connected to the board, then the command returns “failed” message.

3. MDBBillEnable

GUI command	
Command	**Parameters/Comments**
MDBBillEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillEnable":"success"}
or
{"MDBBillEnable":"failed"}
	This command will activate the attached MDB bill validator. This command must be preceded by the MDBBillInit command. You cannot activate a bill validator if this one is not initialized.

4. MDBBillDisable

GUI command	
Command	**Parameters/Comments**
MDBBillDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillDisable":"success"}
or
{"MDBBillDisable":"failed"}
	This command will deactivate the attached MDB bill validator.

5. MDBCoinInit

GUI command	
Command	**Parameters/Comments**
MDBCoinInit	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCoinInit":"success"}
or
{"MDBCoinInit":"failed"}
	This command will initialize the attached MDB coin acceptor/changer.

6. MDBCoinEnable

GUI command	
Command	**Parameters/Comments**
MDBCoinEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCoinEnable":"success"}
or
{"MDBCoinEnable":"failed"}
	This command will enable the attached MDB coin acceptor/changer. This command requires a previous MDBCoinInit command. You cannot enable a coin acceptor/changer if it was not previously initialized.

7. MDBCoinDisable

GUI command	
Command	**Parameters/Comments**
MDBCoinDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinDisable":"success"} or {"MDBCoinDisable":"failed"}	This command will disable the attached MDB coin acceptor/changer.

8. MDBBillSettings

GUI command	
Command	**Parameters/Comments**
MDBBillSettings	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{ "Type":"BillSettings", "level":"1", "CurrencyCode":"1642", "ScalingFactor":"100", "DecimalPlaces":"2", "StackerCapacity":"300", "EscrowAvailable":"true", "BillValues":[{"BillValue":"1"}, {"BillValue":"5"}, {"BillValue":"10"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}, {"BillValue":"0"}], "Manufacturer":"ITL", "SerialNumber":"000051005569", "Model":"BV0020 000", "SWVersion":"1042" }	Using this command you can get informations about the MDB connected bill validator. <BillValues> array contains the bill values that the device will recognize. Most of those are low level informations and you do not need to use them in you application, since all MDB protocol will be handled by the RASPIVEND board. You can use, for example, the serial number and model to keep tracking of the payment systems, service activities, cleaning, etc.

9. MDBCoinSettings

GUI command	
Command	**Parameters/Comments**
MDBCoinSettings	[none]

RASPIVEND daemon answer	
Answer	**Parameters/Comments**

```
{
  "Type":"CoinSettings",
  "Level":"3",
  "CurrencyCode":"0040",
  "ScalingFactor":"10",
  "DecimalPlaces":"2",
  "CoinValues":[
    {"CoinValue":"1"},
    {"CoinValue":"5"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"},
    {"CoinValue":"0"}
  ],
  "ChangeCoins":[
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"true"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"},
    {"ChangeCoin":"false"}
  ],
  "Manufacturer":"NRI",
  "SerialNumber":"NRI00136295-",
  "Model":"046G-46.F400",
  "SWVersion":"12320"
}
```

Using this command you can get informations about the MDB connected coin acceptors/changers.

<CoinValues> array contains the values that the device will recognize.
<ChangeCoins> array contains informations about the coins that the device can use to return change. If all elements are “false”, then the device does not have change capabilities.

Most of those are low level informations and you do not need to use them in you application, since all MDB protocol will be handled by the RASPIVEND board.

10. MDBSetMaxCredit

GUI command	
Command	**Parameters/Comments**
MDBSetMaxCredit(“NNNN”)	Maximum MDB credit accepted - 32 bit positive value When there is a bill in escrow, where <current_MDB_Credit> + <MDB_escrow_bill_value> is bigger than NNNN, then that bill wil be returned to the customer. For coins, your application must disable the coin acceptor/changer, once the maximum credit is reached. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetMaxCredit":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetMaxCreditOK, after you verify that the returned value is the same with the sent value.

11. MDBSetMaxCreditOK

GUI command	
Command	**Parameters/Comments**
MDBSetMaxCreditOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetMaxCreditOK":"success"}
or
{"MDBSetMaxCreditOK":"failed"}
	This command will commit the change of the MaxCredit variable on the RASPIVEND board.

12. MDBSetCurrentCredit

GUI command	
Command	**Parameters/Comments**
MDBSetCurrentCredit(“NNNN”)	Set the value of the MDB module curernt credit. - 32 bit positive value This value must be set after each succesful vend and before issuing the “MDBCashlessRevalue” command. This is the value that the RASPIVEND board will try to use for revalue. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100). This command will modify the value of CurrentCreditCash. You can read this variable by issuing the MDBPoll command.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetCurrentCredit":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetCurrentCreditOK, after you verify that the returned value is the same with the sent value.

13. MDBSetCurrentCreditOK

GUI command	
Command	**Parameters/Comments**
MDBSetCurrentCreditOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetCurrentCreditOK":"success"}
or
{"MDBSetCurrentCreditOK":"failed"}
	This command will commit the change of the CurrentCreditCash variable on the RASPIVEND board.

14. MDBSetChange

GUI command	
Command	**Parameters/Comments**
MDBSetChange("NNNN")	Set the value of the change that MDB module will return when the MDBSetChangeOK command will be issued. - 32 bit positive value This value must be set before any change return action The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetChange":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetChangeOK, after you verify that the returned value is the same with the sent value.

15. MDBSetChangeOK

GUI command	
Command	**Parameters/Comments**
MDBSetChangeOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetChangeOK":"success"}
or
{"MDBSetChangeOK":"failed"}
	This command will commit the MDBSetChange command and will try to return the requested amount to the customer.
	You need to check the changer status by repeatedly issuing the MDBPoll command and parsing the changer status variable. This variable will be described on MDBPoll section.

16. MDBCreditReset

GUI command	
Command	**Parameters/Comments**
MDBCreditReset	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCreditReset":"success"}
or
{"MDBCreditReset":"failed"}
	This command will set the CurerntCreditCash variable to 0.
	It is recommended to use this command before activating the payment systems on each transaction.

17. MDBPoll

GUI command	
Command	**Parameters/Comments**
MDBPoll	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"Poll",
"CurrentCreditCash":"0",
"CurrentCreditCashless":"0",
"BillStat":"0B000B00",
"CoinStat":"00000200",
"CashlessStat":"FFFFFFFF",
"CashlessID":"FFFFFFFF"
}
	This command will return the current MDB module's information.
	It is recommended to issue this command at least twice a second while the transaction is opened (when the payment systems are enabled), to check the current credit and to disable the payment systems when the credit is the same or bigger than the selected product price.
	- <CurrentCreditCash> is the accumulated cash credit;
	- <CurrentCreditCashless> is the credit available on the customer's cashless payment media;
	- <BilStat> contains the last 4 hexadecimal bill validator status codes (according with the Table 2 on page 14). The leftmost value is the older one.
	- <CoinStat> contains the last 4 hexadecimal coin acceptor status codes (according with the Table 3 on page 15). The leftmost value is the older one.
	- <CaslessStat> contains the last 4 hexadecimal cashless devices status codes (according with the Table 4 on page 16). The leftmost value is the older one.
	- <CashlessID> contains the internal ID of the customer's media. You can use this for tracking purposes.
	It is recommended to keep an eye on this informations during the transaction. When there is no transaction open it is recommended to periodically poll this status variable to detect payment systems jam.
	If a payment system was not initialized, it's corresponding status variable will have “FFFFFFFF” value.

18. MDBResetStatus

GUI command	
Command	**Parameters/Comments**
MDBResetStatus	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBResetStatus":"success"}
or
{"MDBResetStatus":"failed"}
	This command will set to “00000000” all MDB payment system status variables.
	If a payment system was not initialized, it's corresponding status variable will have “FFFFFFFF” value.

19. MDBCashlessInit

GUI command	
Command	**Parameters/Comments**
MDBCashlessInit(N)	“N” is the address of the cashless to be initialized (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessInit":"success"}
or
{"MDBCashlessInit":"failed"}
	This command will perform all initialization tasks for the attached MDB cashless system. If something goes wrong or the MDB cashless system is not connected to the board, then the command returns “failed” message.

20. MDBCashlessEnable

GUI command	
Command	**Parameters/Comments**
MDBCashlessEnable(N)	“N” is the address of the cashless to be enabled (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessEnable":"success"}
or
{"MDBCashlessEnable":"failed"}
	This command will activate the attached MDB cashless system. This command must be preceded by the MDBCashlessInit command. You cannot activate a cashless system if this one is not initialized.

21. MDBCashlessDisable

GUI command	
Command	**Parameters/Comments**
MDBCashlessDisable(N)	“N” is the address of the cashless to be disabled (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessDisable":"success"}
or
{"MDBCashlessDisable":"failed"}
	This command will deactivate the attached MDB cashless system.

22. MDBCashlessSettings

GUI command	
Command	**Parameters/Comments**
MDBCashlessSettings(N)	“N” is the address of the cashless to be readed (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"Cashless",
"level":"2",
"CurrencyCode":"1978",
"ScalingFactor":"1",
"DecimalPlaces":"2",
"Manufacturer":"COM",
"SerialNumber":"000000114761",
"Model":"NEW_EUROKEY ",
"SWVersion":"513"
}
	This command will return the specified cashless device low level informations. You will probably don't need those informations, unless you want to track the payment systems for service/maintenance reasons.

23. MDBCashlessVendRequest

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendRequest(NNNN)	Set the value of the requested credit will be withdrawn from the customer's cashless media.. - 32 bit positive value This value must be set before any MDBCashlessVendRequestOK command. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessVendRequest":"100"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBCashlessVendRequestOK, after you verify that the returned value is the same with the sent value.

24. MDBCashlessVendRequestOK

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendRequestOK(N)	“N” is the address of the cashless to request vend permission(1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendRequestOK":"success"}
or
{"MDBCashlessVendRequestOK":"failed"}
	This command will commit the MDBCashlessVendRequest command.
	You must use the MDBPoll command to verify the response of the cashless system (vend approved or vend denied) according to Table 4 on page 16. If the status changes to “vend approved” then you have to issue the command “MDBCashlessVendSuccess” to withrawal the MDBCashlessVendRequest value from the customer's cashless payment media.

25. MDBCashlessVendSuccess

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendSuccess(N)	“N” is the address of the cashless to withdraw (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendSuccess":"success"}
or
{"MDBCashlessVendSuccess":"failed"}
	This command will send transaction success information to the cashless system.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).

26. MDBCashlessVendFailed

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendFailed(N)	“N” is the address of the cashless to notify that the transaction has failed (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendFailed":"success"}
or
{"MDBCashlessVendFailed":"failed"}
	This command will send transaction failed information to the cashless system.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).

27. MDBCashlessRevalue

GUI command	
Command	**Parameters/Comments**
MDBCashlessRevalue(N)	“N” is the address of the cashless to revalue (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessRevalue":"success"}
or
{"MDBCashlessRevalue":"failed"}
	This command will send load the CurentCreditCash value to the cashless.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).
	Also, your application must handle the maximum revalue settings for the cashless system. You have to set the cashless revalue only in it's aloud range. If you will not manage this, then the cashless can randomly goes to overflow.

28. MDBCashlessEndSession

GUI command	
Command	**Parameters/Comments**
MDBCashlessEndSession(N)	“N” is the address of the cashless to close the session (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessEndSession":"success"}
or
{"MDBCashlessEndSession":"failed"}
	This command will force the cashless system to close the current session. If the media is not removed, most of the time, the cashless system will automatically open a new session.

29. KBSet + KBSetOK

GUI command	
Command	**Parameters/Comments**
KBSet(ABCD)	This will send a command to the VTLCOMBUS extension to simulate a keypress to the vending machine. For the moment, the VTLCOMBUS is only available for Necta and Wurlitzer vending machines. We can produce any other keyboard extensions on request. - A is the the block number; - B is the first block multiplexer channel control line - C is the second block multiplexer channel control line - D is the third block multiplexer channel control line A full list with key correspondence will be delivered on request, depending on you machine's manufacturer and model. This command will send the appropriate keystroke to deliver the product. When you receive the correct combination as a response, then you have to commit the command with the command **KBSetOK**
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"KeyboardSet":"ABCD"}
or
{"KeyboartSet":"failed"}
	This command will simulate a keyboard press to deliver one product to the customer.

Keyboard truth table shows the connections between lines and columns, according to the selected bock and multiplexer values (external keyboard simulator needed, sold separately).

A	B	C	D	Connection (VMCR = KB row, VMCC = KB col)
1	0	0	0	VMCR3 ↔ VMCC1
1	0	0	1	VMCR2 ↔ VMCC1
1	0	1	0	VMCR1 ↔ VMCC1
1	0	1	1	VMCR4 ↔ VMCC1
1	1	0	0	VMCR5 ↔ VMCC1
1	1	0	1	VMCR2 ↔ VMCC2
1	1	1	0	VMCR1 ↔ VMCC2
1	1	1	1	Not used
2	0	0	0	VMCR3 ↔ VMCC2
2	0	0	1	VMCR5 ↔ VMCC2
2	0	1	0	VMCR4 ↔ VMCC2
2	0	1	1	VMCR2 ↔ VMCC3
2	1	0	0	VMCR1 ↔ VMCC3
2	1	0	1	VMCR4 ↔ VMCC3
2	1	1	0	VMCR3 ↔ VMCC3
3	0	0	0	VMCR5 ↔ VMCC3

30. RTCSet

GUI command	
Command	**Parameters/Comments**
RTCSet	This command will store the current Rasberry PI date and time to the RASPIVEND's non-volatile RTC.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"RTCSet”:”success”}
or
{"RTCSet":"failed"}	

31. RTCGet

GUI command	
Command	**Parameters/Comments**
RTCGet	This command will read the date and time informations from the non-volatile RTC on the RASPIVEND board. This command is useful to help you set the date and time on Raspberry PI if it is not connected to the Internet.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"RTCValue",
"hour":"19",
"minute":"42",
"second":"47",
"day":"28",
"month":"0 3",
"year":"2016"
}	

V. High level mode with Python 3 service daemon

Using this mode, the development becomes much easier. The communication with the board and the peripherals is managed by a small Python 3 application, called pyraspivend.py. The application is available for download on product's page.
Commands are not case sensitive. We have used capitalization to facilitate reading.
It is a good idea for your application to retry sending the command few times if you get an "failed" answer. This answer can be returned in the event of Raspberry PI to RASPIVEND board communication failure. Also it can be returned if you try to address a not connected device.

1. Configuring Python 3

The only dependency for this application is pyserial library, version 3.0.1 or higher.

2. MDBBillInit

GUI command	
Command	**Parameters/Comments**
MDBBillInit	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillInit":"success"}
or
{"MDBBillInit":"failed"}
	This command will perform all initialization tasks for the attached MDB bill validator. If something goes wrong or the MDB bill validator is not connected to the board, then the command returns "failed" message.

3. MDBBillEnable

GUI command	
Command	**Parameters/Comments**
MDBBillEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillEnable":"success"}
or
{"MDBBillEnable":"failed"}
	This command will activate the attached MDB bill validator. This command must be preceded by the MDBBillInit command. You cannot activate a bill validator if this one is not initialized.

4. MDBBillDisable

GUI command	
Command	**Parameters/Comments**
MDBBillDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBBillDisable":"success"}
or
{"MDBBillDisable":"failed"}
	This command will deactivate the attached MDB bill validator.

5. MDBCoinInit

GUI command	
Command	**Parameters/Comments**
MDBCoinInit	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCoinInit":"success"}
or
{"MDBCoinInit":"failed"}
	This command will initialize the attached MDB coin acceptor/changer.

6. MDBCoinEnable

GUI command	
Command	**Parameters/Comments**
MDBCoinEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCoinEnable":"success"}
or
{"MDBCoinEnable":"failed"}
	This command will enable the attached MDB coin acceptor/changer. This command requires a previous MDBCoinInit command. You cannot enable a coin acceptor/changer if it was not previously initialized.

7. MDBCoinDisable

GUI command	
Command	**Parameters/Comments**
MDBCoinDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCoinDisable":"success"}
or
{"MDBCoinDisable":"failed"}
	This command will disable the attached MDB coin acceptor/changer.

8. MDBBillSettings

GUI command	
Command	**Parameters/Comments**
MDBBillSettings	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"BillSettings",
"level":"1",
"CurrencyCode":"1642",
"ScalingFactor":"100",
"DecimalPlaces":"2",
"StackerCapacity":"300",
"EscrowAvailable":"true",
"BillValues":[
{"BillValue":"1"},
{"BillValue":"5"},
{"BillValue":"10"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"},
{"BillValue":"0"}
],
"Manufacturer":"ITL",
"SerialNumber":"000051005569",
"Model":"BV0020   000",
"SWVersion":"1042"
}
	Using this command you can get informations about the MDB connected bill validator.
	
	<BillValues> array contains the bill values that the device will recognize.
	
	Most of those are low level informations and you do not need to use them in you application, since all MDB protocol will be handled by the RASPIVEND board.
	
	You can use, for example, the serial number and model to keep tracking of the payment systems, service activities, cleaning, etc.

9. MDBCoinSettings

GUI command	
Command	**Parameters/Comments**
MDBCoinSettings	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{ "Type":"CoinSettings", "Level":"3", "CurrencyCode":"0040", "ScalingFactor":"10", "DecimalPlaces":"2", "CoinValues":[{"CoinValue":"1"}, {"CoinValue":"5"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}, {"CoinValue":"0"}], "ChangeCoins":[{"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"true"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}, {"ChangeCoin":"false"}], "Manufacturer":"NRI", "SerialNumber":"NRI00136295-", "Model":"046G-46.F400", "SWVersion":"12320" }	Using this command you can get informations about the MDB connected coin acceptors/changers. <CoinValues> array contains the values that the device will recognize. <ChangeCoins> array contains informations about the coins that the device can use to return change. If all elements are “false”, then the device does not have change capabilities. Most of those are low level informations and you do not need to use them in you application, since all MDB protocol will be handled by the RASPIVEND board.

10. MDBSetMaxCredit

GUI command	
Command	**Parameters/Comments**
MDBSetMaxCredit(“NNNN”)	Maximum MDB credit accepted - 32 bit positive value When there is a bill in escrow, where <current_MDB_Credit> + <MDB_escrow_bill_value> is bigger than NNNN, then that bill wil be returned to the customer. For coins, your application must disable the coin acceptor/changer, once the maximum credit is reached. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetMaxCredit":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetMaxCreditOK, after you verify that the returned value is the same with the sent value.

11. MDBSetMaxCreditOK

GUI command	
Command	**Parameters/Comments**
MDBSetMaxCreditOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetMaxCreditOK":"success"}
or
{"MDBSetMaxCreditOK":"failed"}
	This command will commit the change of the MaxCredit variable on the RASPIVEND board.

12. MDBSetCurrentCredit

GUI command	
Command	**Parameters/Comments**
MDBSetCurrentCredit(“NNNN”)	Set the value of the MDB module curernt credit. - 32 bit positive value This value must be set after each succesful vend and before issuing the “MDBCashlessRevalue” command. This is the value that the RASPIVEND board will try to use for revalue. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100). This command will modify the value of CurrentCreditCash. You can read this variable by issuing the MDBPoll command.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetCurrentCredit":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetCurrentCreditOK, after you verify that the returned value is the same with the sent value.

13. MDBSetCurrentCreditOK

GUI command	
Command	**Parameters/Comments**
MDBSetCurrentCreditOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetCurrentCreditOK":"success"}
or
{"MDBSetCurrentCreditOK":"failed"}
	This command will commit the change of the CurrentCreditCash variable on the RASPIVEND board.

14. MDBSetChange

GUI command	
Command	**Parameters/Comments**
MDBSetChange(“NNNN”)	Set the value of the change that MDB module will return when the MDBSetChangeOK command will be issued. - 32 bit positive value This value must be set before any change return action The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBSetChange":"NNNN"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBSetChangeOK, after you verify that the returned value is the same with the sent value.

15. MDBSetChangeOK

GUI command	
Command	**Parameters/Comments**
MDBSetChangeOK	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBSetChangeOK":"success"}
or
{"MDBSetChangeOK":"failed"}
	This command will commit the MDBSetChange command and will try to return the requested amount to the customer.
	You need to check the changer status by repeatedly issuing the MDBPoll command and parsing the changer status variable. This variable will be described on MDBPoll section.

16. MDBCreditReset

GUI command	
Command	**Parameters/Comments**
MDBCreditReset	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCreditReset":"success"}
or
{"MDBCreditReset":"failed"}
	This command will set the CurerntCreditCash variable to 0.
	It is recommended to use this command before activating the payment systems on each transaction.

17. MDBPoll

GUI command	
Command	**Parameters/Comments**
MDBPoll	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"Poll",
"CurrentCreditCash":"0",
"CurrentCreditCashless":"0",
"BillStat":"0B000B00",
"CoinStat":"00000200",
"CashlessStat":"FFFFFFFF",
"CashlessID":"FFFFFFFF"
}
	This command will return the current MDB module's information.
	It is recommended to issue this command at least twice a second while the transaction is opened (when the payment systems are enabled), to check the current credit and to disable the payment systems when the credit is the same or bigger than the selected product price.
	- <CurrentCreditCash> is the accumulated cash credit;
	- <CurrentCreditCashless> is the credit available on the customer's cashless payment media;
	- <BilStat> contains the last 4 hexadecimal bill validator status codes (according with the Table 2 on page 14). The leftmost value is the older one.
	- <CoinStat> contains the last 4 hexadecimal coin acceptor status codes (according with the Table 3 on page 15). The leftmost value is the older one.
	- <CaslessStat> contains the last 4 hexadecimal cashless devices status codes (according with the Table 4 on page 16). The leftmost value is the older one.
	- <CashlessID> contains the internal ID of the customer's media. You can use this for tracking purposes.
	It is recommended to keep an eye on this informations during the transaction. When there is no transaction open it is recommended to periodically poll this status variable to detect payment systems jam.
	If a payment system was not initialized, it's corresponding status variable will have “FFFFFFFF” value.

18. MDBResetStatus

GUI command	
Command	**Parameters/Comments**
MDBResetStatus	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBResetStatus":"success"}
or
{"MDBResetStatus":"failed"}
	This command will set to “00000000” all MDB payment system status variables.
	If a payment system was not initialized, it's corresponding status variable will have “FFFFFFFF” value.

19. MDBCashlessInit

GUI command	
Command	**Parameters/Comments**
MDBCashlessInit(N)	“N” is the address of the cashless to be initialized (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessInit":"success"}
or
{"MDBCashlessInit":"failed"}
	This command will perform all initialization tasks for the attached MDB cashless system. If something goes wrong or the MDB cashless system is not connected to the board, then the command returns “failed” message.

20. MDBCashlessEnable

GUI command	
Command	**Parameters/Comments**
MDBCashlessEnable(N)	“N” is the address of the cashless to be enabled (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessEnable":"success"}
or
{"MDBCashlessEnable":"failed"}
	This command will activate the attached MDB cashless system. This command must be preceded by the MDBCashlessInit command. You cannot activate a cashless system if this one is not initialized.

21. MDBCashlessDisable

GUI command	
Command	**Parameters/Comments**
MDBCashlessDisable(N)	“N” is the address of the cashless to be disabled (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessDisable":"success"}
or
{"MDBCashlessDisable":"failed"}
	This command will deactivate the attached MDB cashless system.

22. MDBCashlessSettings

GUI command	
Command	**Parameters/Comments**
MDBCashlessSettings(N)	“N” is the address of the cashless to be readed (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{
"Type":"Cashless",
"level":"2",
"CurrencyCode":"1978",
"ScalingFactor":"1",
"DecimalPlaces":"2",
"Manufacturer":"COM",
"SerialNumber":"000000114761",
"Model":"NEW_EUROKEY ",
"SWVersion":"513"
}
	This command will return the specified cashless device low level informations. You will probably don't need those informations, unless you want to track the payment systems for service/maintenance reasons.

23. MDBCashlessVendRequest

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendRequest(NNNN)	Set the value of the requested credit will be withdrawn from the customer's cashless media.. - 32 bit positive value This value must be set before any MDBCashlessVendRequestOK command. The value is multiplied by the 100 scaling factor (for example, for 1EUR you have to set this value to 100).
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessVendRequest":"100"}	This command will return the value you have sent. For safety reasons (eliminate the communication errors), you must commit this action with MDBCashlessVendRequestOK, after you verify that the returned value is the same with the sent value.

24. MDBCashlessVendRequestOK

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendRequestOK(N)	“N” is the address of the cashless to request vend permission(1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendRequestOK":"success"}
or
{"MDBCashlessVendRequestOK":"failed"}
	This command will commit the MDBCashlessVendRequest command.
	You must use the MDBPoll command to verify the response of the cashless system (vend approved or vend denied) according to Table 4 on page 16. If the status changes to “vend approved” then you have to issue the command “MDBCashlessVendSuccess” to withrawal the MDBCashlessVendRequest value from the customer's cashless payment media.

25. MDBCashlessVendSuccess

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendSuccess(N)	“N” is the address of the cashless to withdraw (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendSuccess":"success"}
or
{"MDBCashlessVendSuccess":"failed"}
	This command will send transaction success information to the cashless system.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).

26. MDBCashlessVendFailed

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendFailed(N)	“N” is the address of the cashless to notify that the transaction has failed (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessVendFailed":"success"}
or
{"MDBCashlessVendFailed":"failed"}
	This command will send transaction failed information to the cashless system.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).

27. MDBCashlessRevalue

GUI command	
Command	**Parameters/Comments**
MDBCashlessRevalue(N)	"N" is the address of the cashless to revalue (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessRevalue":"success"}
or
{"MDBCashlessRevalue":"failed"}
	This command will send load the CurentCreditCash value to the cashless.
	You must use the MDBPoll command to verify the response of the cashless system according to Table 4 on page 16).
	Also, your application must handle the maximum revalue settings for the cashless system. You have to set the cashless revalue only in it's aloud range. If you will not manage this, then the cashless can randomly goes to overflow.

28. MDBCashlessEndSession

GUI command	
Command	**Parameters/Comments**
MDBCashlessEndSession(N)	"N" is the address of the cashless to close the session (1 for the first cashless device and 2 for the second cashless device)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"MDBCashlessEndSession":"success"}
or
{"MDBCashlessEndSession":"failed"}
	This command will force the cashless system to close the current session. If the media is not removed, most of the time, the cashless system will automatically open a new session.

29. SetMUXChannel

GUI command	
Command	**Parameters/Comments**
SetMUXChannel("channel")	This commend selects the cnannel for communication. "channel" can take the following values: - RS232_1 - RS232_2 - VTLCOMBUS - ccTalk
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{"SetMUXChannel":"channel"}
or
{"SetMUXChannel":"failed"}
	This command will switch the board multiplexer and select the channel connected to the Raspbery PI serial port.

30. CCTHopperInit

GUI command	
Command	**Parameters/Comments**
CCTHopperInit(N)	“N” is the address of the hopper you need to perform initialization. The address depends on the hoper settings.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{ "HopperInitAddress" : "3" , "Reset" : "success" , "ClearComms" : "failed" , "RequestStatus" : "success" , "ManufacturerID" : "CPS" , "BuildCode" : "Combo" , "EquipmentCategory" : "Payout" , "HiLevelSensor" : "present" , "LoLevelSensor" : "present" , "HiLevelSensor" : "0" , "LoLevelSensor" : "0" , "HopperEnable" : "success" }	This command will perform a full init of the hopper with provided address. The response include the status for each stage in the initialization procedure. Your application must decide, based on those statuses, if the init procedure is a success or not. In our example, the procedure is a success, since the test hopper does not support “ClearComms” command, but successfully answered on the most important init procedures and also to the “HopperEnable” command.

31. CCTHopperDispense

GUI command	
Command	**Parameters/Comments**
CCTHopperDispense(N,M)	“N” is the hopper address and “M” is the number of coins to dispense.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{“HopperDispenseNormal":"success"}
or
{“HopperDispenseNormal":"failed"}
	This command will instruct the hopper to payout a number of coins. After issuing this command, the customer’s application should keep polling with command CCTHopperCheckDispense, to get informations about the status, number of coins dispensed, number of coins to dispense and error.

32. CCTHopperCheckDispense

GUI command	
Command	**Parameters/Comments**
CCTHopperCheckDispense(N)	“N” is the hopper address.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
Ex. 1 – success { "EventCounter" : "1" , "ToDispense" : "0" , "Dispensed" : "2" , "NotDispensed" : "0" } Ex. 2 - failed { "EventCounter" : "4" , "ToDispense" : "0" , "Dispensed" : "2" , "NotDispensed" : "8" } Ex. 3 – dispense in progress { "EventCounter" : "6" , "ToDispense" : "6" , "Dispensed" : "4" , "NotDispensed" : "0" }	This command will return a message regarding the last known hopper status. After issuing CCTHopperDispense or CCTHopperDispenseCipher commands, the customer’s application should periodically poll the hopper with this command until one of the following situation appears: - “**Dispensed**” has the same value as the number of coins requested. The dispense is fine and the transaction is finished. - “**NotDispensed**” has a value bigger than 0. In this situation, the dispense failed and there a remaining “NotDispensed” coins that could not be supplied. It is up to the customer’s application to decide if it will try another dispense command or return to the main loop.

33. CCTHopperDispenseCipher

GUI command	
Command	**Parameters/Comments**
CCTHopperDispenseCipher(N,M)	“N” is the hopper address and “M” is the number of coins to dispense using cipher key instead of serial number. Cipher key is required by some hoppers instead of plain 3 bytes serial number. Try to use this in case your hopper is correctly initialized but is not dispensing any coin.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**

{“HopperDispenseCipher":"success"}
or
{“HopperDispenseCipher":"failed"}
	This command will instruct the hopper to payout a number of coins. After issuing this command, the customer’s application should keep polling with command CCTHopperCheckDispense, to get informations about the status, number of coins dispensed, number of coins to dispense and error.

NOTES: