Raspberry PI
vending solution
(RASPIVEND DIRECT)
v21.03.2018
Quick Reference

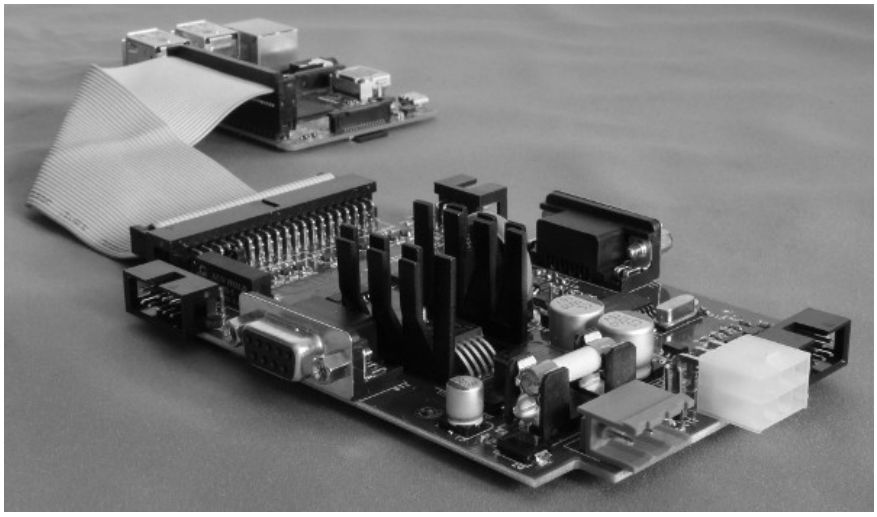


Table of Contents

I. General informations

1. Terms

- **RASPIVEND** = Raspberry PI vending board (shield).
- **RASPIVEND DAEMON –** Python 3 based RASPIVEND DIRECT management daemon.
- **MDB PERIPHERALS** = payment systems connected on the MDB bus.
- **CCTALK PERIPHERALS** = payment systems and peripherals, connectod to ccTalk bus.
- **HOST APPLICATION** = the Python 3 daemon
- **CLIENT APLICATION** = the client application that will connect to the socket of the HOST APPLICATION
- **ACK** = acknowledge
- **NACK** = not-acknowledge

2. Differences from the RASPIVEND version

- This version is totally different, since it contains a firmware that only manages the MDB low level ACK/NACK messages, the critical 5ms timing and making a translation from 8bit to 9bit communication (and back from 9bit to 8bit, automatically handling the mode bit). The whole logic is moved to the high level, on to the Raspberry Pi. The user can send any MDB command to on the serial port and will get all the responses back, without modification. Practically there are 2 possibilities:

1. The users are implementing the needed MDB sections in their applications (sending any MDB message directly to the serial port and also implementing the poll for the connected payment systems).

2. The users are using our daemon to communicate with the board through simple messages that are returning JSON responses. The daemon implements a Level 2 VMC controller. Also, the daemon supports raw message communication, with or without automatic MDB CRC calculation.

3. 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, the user application is responsible of all VMC logic and payment systems manipulation along with multiplexer handling.

B. High level mode

In high level mode, the user's app is connecting using sockets to the 5127 TCP port on localhost and sends some standard messages, described below. The are are no access to multiplexer pins and only implemented and describe functions are available.

4. Low level 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	115200
data bits	8
parity	NONE
hardware flow	YES (RTS/CTS)
software flow	NO

Table 3: MDB communication parameters

Scenario #1 - If you are using Raspbian on Raspberry Pi 3 and you don't need Bluetooth, you have to redirect the /dev/ttyAMA0 serial port back to the 40pin connector, using the following procedure:

Using your favorite text editor, open /boot/config.txt and add the following 2 lines:

```
dtoverlay=pi3-disable-bt
core_freq=250
enable_uart=1
```

Using your favorite text editor, open /boot/cmdline.txt, search for the following settings and delete them:

```
console=serial0,115200
console=ttyAMA0,115200
```

Reboot your Pi.
You will not able to use RPi Bluetooth with those settings.

Scenario #2 – If you are using Raspbian on Raspberry Pi 3 and you need also to use Bluetooth.

Using your favorite text editor, open /boot/config.txt and add the following 2 lines:

```
#dtoverlay=pi3-disable-bt
core_freq=250
enable_uart=1
```

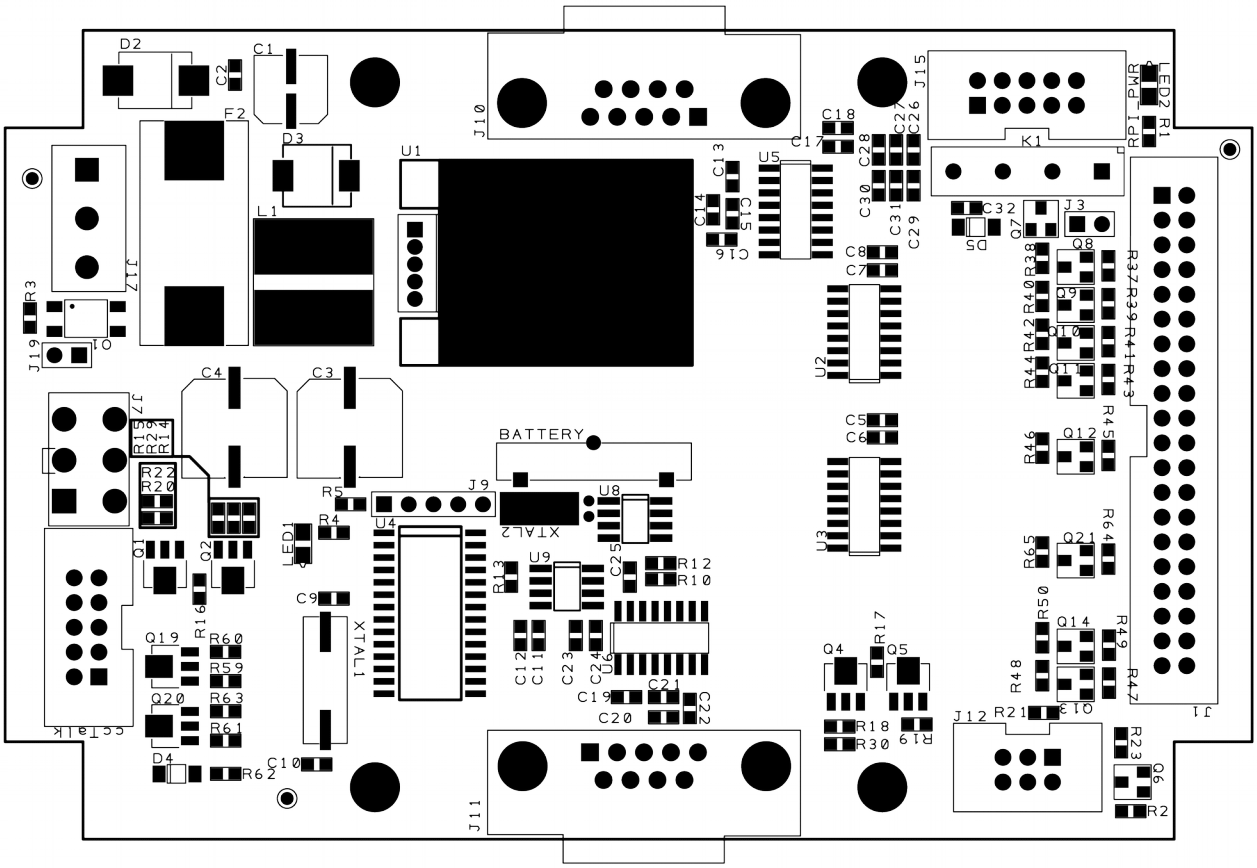
Using your favorite text editor, open /boot/cmdline.txt, search for the following settings and delete them:

```
console=serial0,115200
console=ttyAMA0,115200
```

Reboot your Pi.
Modify the main .py demo file search for /dev/ttyAMA0 and change to /dev/serial0.
You will be able to use both Bluetooth and RASPISLAVE board.

This device was tested with Raspberry Pi 2, Raspberry Pi 3 Model B v2.0, Raspberry Pi 3 B+ and Raspberry Pi Zero W.

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 DC 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. Connect pin #1 (squared pin) to VDC and pin #2 to GND. Pin #3 is not used in this configuration.
- **<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

The user is responsible to send any correct MDB message to the interface that will translate it to 9bit, handling the mode bit. Also, any answer from the payment systems will be translated to 8bit format and sent back to the serial port.

For settings RTC, you need to send to following message:

RTC set - <0xFE> <0x01> <ss> <mm> <hh> <dow> <dd> <MM> <yy> <CRC>

Device answer will be 0xFC 0xFC 0xFC 0xFC 0xFC 0xFC on success and timeout on failure.

RTC get - <0xFE> <0x02> <CRC>

Device answer will be in the following format:

<0xFE> <0x02> <ss> <mm> <hh> <dow> <dd> <MM> <yy> <CRC>

All parameters on RTC set and get are BCD (please check the DS1307 IC manual on page 8 to find registers' interpretation).

CRC calculation follows the standard MDB rules for both MDB messages and RTC manipulation messages.

IV. High level mode with service daemon

Using this mode, the development becomes much easier. The communication with the board and the peripherals is managed by a small application, called pyraspivend_direct.. 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. Since on Raspberry Pi 3 the UART interface is routed to Bluetooth, you must redirect it again to 40pin connector. Also, to successfully run the application, you have to copy it into a directory and decompress the archive, keeping the shared libraries in the same place with the main application. Also, it requires some Python 3 libraries, so it is highly recommended to install Python 3 on your Raspberry Pi.

1. Configuring Python 3

To use this mode, you need the following:

- install Python 3 on your Raspberry Pi (for Raspbian “sudo apt-get install python3”);
- install pip3 (for Raspbian “sudo apt-get install python3-pip”);
- install PySerial (“sudo pip3 install pyserial==3.0.1”);
- download and run the Python script from our website (using python3 interpreter);
- open a new console and run telnet on localhost, port 5127 (“sudo telnet localhost 5127”)
- in the telnet window start sending commands to the device.

2. BillReset

GUI command	
Command	**Parameters/Comments**
BillReset	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIIIReset": 0} or {"MDBBIIIReset": -1}	This command will send the reset command to the bill validator. If the returned value is “0”, the command were successfully executed and the validator responded with ACK. If the returned value is “-1”, there was an error reseting the bill validator and the user’s application should retry few times before aborting the operation. When you have a connected bill validator, this is the first command to send in the initialization process

3. BillInit

GUI command	
Command	**Parameters/Comments**
BillInit	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIIIReset": 0} or {"MDBBIIIReset": -1}	This command will perform the initialization procedure on the bill validator. The answers could be “0” - success or “-1” - failed. When you have a connected bill validator, this is the second command to send in the initialization process

4. BillSettings

GUI command	
Command	**Parameters/Comments**
BillSettings	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBillSettings": "Current", "Level": 2, "CountryCode": 1642, "ScalingFactor": 100, "StackerCappacity": 300, "EscrowAvailable": true, "BillValues":[1,5,10,0,0,0,0,0,0,0,0,0,0,0,0,0], "Manufacturer": "ITL", "SerialNumber": "000000271269", "Model": "BV0100 000", "SoftwareVersion": "0414", "RecyclingAvaliable": false}	This command will return all validator's settings When you have a connected bill validator, this is the third command to send in the initialization process. This command is mandatory, otherwise the application will not be able to perform some calculations (for example, bill values) because it has not enough informations (for example, scaling factor or decimal places, plus the accepted bills values)

5. BillStacker

GUI command	
Command	**Parameters/Comments**
BillStacker	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIllStacker": 38,"StackerFull": false} or {"MDBBIllStacker": -1} if not succeeded	This command will return the number of the bills in stacker (if the bill validator has a stacker and it will return "true" if the stacker is full or "false" if the stacker is not full, yet. Please note that this function depends on the bill validator and some of them may always return 0 and false.

6. BillEnable

GUI command	
Command	**Parameters/Comments**
BillEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIllEnable": 0} – for success or {"MDBBIllEnable": -1} – for failure	This command will enable the attached MDB bill validator. The bill validator will accept all denominations that are supported by it's internal firmware.

7. BillDisable

GUI command	
Command	**Parameters/Comments**
BillDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIllDisable": 0} – for success or {"MDBBIllDisable": -1} – for failure	This command will disable the attached MDB bill validator. It will no longer accept any of the denominations that are supported by it's firmware

8. BillAccept

GUI command	
Command	**Parameters/Comments**
BillAccept	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIllAcceptBillInEscrow": 0} – response if the command was accepted by the bill validator. After this response, the bill will process the bill and try to stack it. When the bill is correctly stacked, the interface will send a second message (unsolicited message): {"BillStacked": 0,"BillValue": 100} that will show the bill is safely deposited in stacker or inside the vending machine (if the bill validator is stackerless). This is the message that should be used to increment the current credit. {"MDBBIllAcceptBillInEscrow": -1} - if the command failed to reach the bill validator	This command can be used only when the bill validator has escrow capability (it gets the bill, recognizes it and keep it in escrow position sending a message about this) The interface will send a message like: {"BillInEscrow": 0,"BillValue": 100} to notify the bill number and the bill value (according to the scaling factor) waiting to accept or reject.

9. BillReject

GUI command	
Command	**Parameters/Comments**
BillReject	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBBIllRejectBillInEscrow": 0} – response if the command is accepted by the bill validator. When the bill is turned to the customer, the interface will send a second message (unsolicited message): {"BillReturned": 0,"BillValue": 100} that will show the bill is correctly returned to the customer. {"MDBBIllRejectBillInEscrow": -1} - if the command failed to reach the bill validator	This command can be used only when the bill validator has escrow capability (it gets the bill, recognizes it and keep it in escrow position sending a message about this) The interface will send a message like: {"BillInEscrow": 0,"BillValue": 100} to notify the bill number and the bill value (according to the scaling factor) waiting to accept or reject.

10. CoinReset

GUI command	
Command	**Parameters/Comments**
CoinReset	
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinReset": 0} – if success or {"MDBCoinReset": -1} – if failed	This command will send the reset command to the coin acceptor. If the returned value is “0”, the command were successfully executed and the coin acceptor responded with ACK. If the returned value is “-1”, there was an error reseting the coin acceptor and the user’s application should retry few times before aborting the operation. When you have a connected coin acceptor, this is the first command to send in the initialization process

11. CoinInit

GUI command	
Command	**Parameters/Comments**
CoinInit	
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinReset": 0} – if success or {"MDBCoinReset": -1} – if failed	This command will perform the initialization procedure on the coin acceptor. The answers could be “0” - success or “-1” - failed. When you have a connected bill validator, this is the second command to send in the initialization process

12. CoinSettings

GUI command	
Command	**Parameters/Comments**
CoinSettings	
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinSettings": "Current", "Level": 3, "CountryCode": 1642, "ScalingFactor": 5, "DecimalPlaces": 2, "CoinRoutingChannel": [0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0], "CoinValues": [1,2,10,255,0,0,0,0,0,0,0,0,0,0,0,0], "Manufacturer": "MEI", "SerialNumber": "2378G802863 ", "Model": "CF7900MDB ", "SoftwareVersion": "0118", "AlternativePayout": true}	This command will return all coin acceptor’s settings. When you have a connected coin acceptor, this is the third command to send in the initialization process. This command is mandatory, otherwise the application will not be able to perform some calculations (for example, coin values) because it has not enough informations (for example, scaling factor or decimal places, plus the accepted coin values)

13. CoinEnable

GUI command	
Command	**Parameters/Comments**
CoinEnable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinEnable": 0} – for success or {"MDBCoinEnable": -1} – for failure	This command will enable the attached MDB coin acceptor. The coin aceptor will accept all denominations that are supported by it’s internal firmware.

14. CoinDisable

GUI command	
Command	**Parameters/Comments**
CoinDisable	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinEnable": 0} – for success or {"MDBCoinEnable": -1} – for failure	This command will disable the attached MDB coin aceptor. It will no longer accept any of the denominations that are supported by it's firmware.

15. CoinTubeStatus

GUI command	
Command	**Parameters/Comments**
CoinTubeStatus	[none]
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinTubeStatus": 13240} –for success (in this example, the total available change is EUR 132.40 or {"MDBCoinTubeStatus": -1} – for failure	This comand will return scaled value of the total change available in tubes, that can be used to return change after transactions

16. CoinChange(NNN)

GUI command	
Command	**Parameters/Comments**
CoinChange(NNN)	"NNN" is the total value of the change that should be returned to the customer. For example, CoinChange(130) will return EUR 1.30
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinChange": 0} – if success This response may be followed by one or more unsolicited messages (depending on the coin acceptor MDB implementation). For example, it can notify that the changer is busy returning change, by: {"CoinStatus": "ChangerPayoutBusy", "CoinStatusCode" : 2} followed by: {"CoinStatus": "OK","CoinStatusCode" : 0} when the changer finished the action.	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. During this command you have to poll the payout status by issuing the CoinPayStatus command (see details on 17.)

17. CoinPayStatus

GUI command	
Command	**Parameters/Comments**
CoinPayStatus	
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCoinChangeStatus": 130} – for success (this example means that EUR1.30 were ejected). or {"MDBCoinChangeStatus": -1} – if failure	This command will return the total value of the ejected coins until the command is issued.

18. CashlessReset(N)

GUI command	
Command	**Parameters/Comments**
CashlessReset(1)	This command will perform a reset of the cashless device number “N”, where “N” can be 1 or 2, depending on your cashless settings.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessReset": 0} – on success or {"MDBCashlessReset": -1} – on failure	

19. CashlessInit(N)

GUI command	
Command	**Parameters/Comments**
CashlessInit(1)	This command will initialize the cashless number “N” where “N” can be 1 or 2, depending on your cashless settings
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessInit": 0} – on success or {"MDBCashlessInit": -1} – on failure	

20. CashlessSettings(N)

GUI command	
Command	**Parameters/Comments**
CashlessSettings(N)	“N” is the cashless number (as deschibed above)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"CashlessLevel": 2, "CashlessCountryCode": 1978, "CashlessScalingFactor": 1, "CashlessDecimalPlaces": 2, "CashlessMaxResponseTime": 7, "CashlessCanRevalue": true, "CashlessCanMultivend": true, "CashlessHasDisplay": false, "CashlessCanCashSale": false, "CashlessManufacturer": "COM", "CashlessSerialNumber": "000000114761", "CashlessModelNumber": "NEW_EUROKEY ", "CashlessSoftwareVersion": 0201}	This command will return all cashless device settings. When you have a connected cashless device, this is the third command to send in the initialization process. This command is mandatory, otherwise the application will not be able to perform some calculations (for example, credit value) because it has not enough informations (for example, scaling factor or decimal places). You may extract some informations here to avoid sending commands that are not supported by the cashless device (for example, revalue if the cashless device does not support it).

21. CashlessEnable(N)

GUI command	
Command	**Parameters/Comments**
CashlessEnable(N)	This command will enable the cashless device number “N”, where “N” can be 1 or 2
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessEnable": 0} – on success or {"MDBCashlessEnable": -1} – on failure	After the cashless is enabled, by using payment media, there could be some unsolicited messages (when a customer swipes the card or inserts the card, toke, etc) {"CashlessNumber": 1, "CashlessStatus": "ReaderBeginSession", "CashlessStatusCode": 3, "CashlessFundsAvailable": 800, "CashlessMediaPaymentId": "0x00 0x53 0x44 0x16", "CashlessPaymentType": "NormalVendCard"} In this example, an RFID kay with EUR8.00 credit was inserted in the reader. The key ID is 0x00 0x53 0x44 0x16 (every payment media has a 4 byte unique ID).

22. CashlessDisable(N)

GUI command	
Command	**Parameters/Comments**
MDBCashlessVendFailed(N)	This command will disable the cashless device number “N”, where “N” can be 1 or 2
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessDisable": 0} – on success or {"MDBCashlessDisable": -1} – on failure	

23. CashlessVendRequest(AAA,BBB,CCC)

GUI command	
Command	**Parameters/Comments**
CashlessVendRequest(AAA,BBB,CCC)	This command can be used only when a session was opened by the cashless device and some credit is reported to the interface - AAA – cashless number (1 or 2); - BBB – product price – the requested product price, scalled by scaling factor) - CCC – product number (for example, 8 is product number 8 on the machine)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessVendRequest": 0} – on success. Other unsolicited messages will appear after this response: {"CashlessNumber": 1, "CashlessStatus": "VendApproved", "CashlessStatusCode": 5, "ApprovedValue": 100} – this means that the cashless device approved the vend and the VMC should dispense the product. The reader will wait for vend result (success or failure) after the transaction is finished Also, the cashless device could respond with the following unsolicited message: {"CashlessNumber": 1, "CashlessStatus": "VendDenied", "CashlessStatusCode": 6} – if the funds ar insufficient for the selected product or for other reasons. {"MDBCashlessVendRequest": -1} – if the command cannot reach the cashless device.	

24. CashlessNegativeVendRequest(AAA,BBB, CCC)

GUI command	
Command	**Parameters/Comments**
CashlessNegativeVendRequest(AAA,BBB, CCC)	This command can be used only when a session was opened by the cashless device and some credit is reported to the interface - AAA – cashless number (1 or 2); - BBB – product price – the requested product price, scalled by scaling factor) - CCC – product number (for example, 8 is product number 8 on the machine). This will add BBB value on the payment media (card, token, etc.) if the vend success is reported by the machine. Do not use this to add credit on the payment media, but only for negative vend transactions (for example, recycling vending machine) and only for cashless payment systems that are supporting negative vend. Use revalue (see below) to add credit on the payment media.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessNegativeVendRequest": 0} – on success. Other unsolicited messages will appear after this response: {"CashlessNumber": 1, "CashlessStatus": "VendApproved", "CashlessStatusCode": 5, "ApprovedValue": 100} – this means that the cashless device approved the vend and the VMC should dispense the product. The reader will wait for vend result (success or failure) after the transaction is finished Also, the cashless device could respond with the following unsolicited message: {"CashlessNumber": 1, "CashlessStatus": "VendDenied", "CashlessStatusCode": 6} – if the funds ar insufficient for the selected product or for other reasons. {"MDBCashlessNegativeVendRequest": -1} – if the command cannot reach the cashless device.	

25. CashlessVendSuccess(AAA,BBB)

GUI command	
Command	**Parameters/Comments**
CashlessVendSuccess(AAA,BBB)	This command will confirm that the transaction is successfully finished on the vending machine. Depending on it's implementation, this is the moment that the cashless device will commit the credit withdrawal. - AAA – the cashless number - BBB – the product number (selection number) dispensed to the customer
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessVendSuccess": 0} – on success or {"MDBCashlessVendSuccess": -1} – on failure	

26. CashlessVendFailed(N)

GUI command	
Command	**Parameters/Comments**
CashlessVendFailed(N)	If the machine fails to dispense the product, then will send this command to the cashless device, to avoid taking the money from the customer's payment media. - "N" is the cashless number
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessVendFailed": 0} – on success or {"MDBCashlessVendFailed": -1} – on failure (cashless payment system did not received the command)	

27. CashlessSessionComplete(N)

GUI command	
Command	**Parameters/Comments**
CashlessSessionComplete(N)	This command will instruct the cashless device “N” to end current session. Eventually, if the payment media was not removed before, the cashless device will open another session. It is a good option to send this command after each finished transaction, to check the cashless available credit.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessSessionComplete": 0} – on success. Depending on the cashless MDB implementation, it will also respond with an unsolicited message of status: {"CashlessNumber": 1, "CashlessStatus": "EndSession", "CashlessStatusCode": 7} or {"MDBCashlessSessionComplete": -1} – on failure	If the payment media was not removed before, some cashless payment systems will open a new session: {"CashlessNumber": 1, "CashlessStatus": "ReaderBeginSession", "CashlessStatusCode": 3, "CashlessFundsAvailable": 600, "CashlessMediaPaymentId": "0x00 0x53 0x44 0x16", "CashlessPaymentType": "NormalVendCard"}

28. CashlessRevalueLimitRequest(N)

GUI command	
Command	**Parameters/Comments**
CashlessRevalueLimitRequest(N)	This command will ask the cashless for the maximum amount that will accept for revalue command (see revalue command below for details)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"CashlessNumber": 1, "CashlessStatus": "LimitAmount", "CashlessStatusCode": 15, "LimitValue": 900} or {"MDBCashlessRevalueLimitRequest": -1} – on failure	The LimitValue will be the maximum value accepted by the cashless device on the next revalue command. If you try to revelue it with a higher value, then it will respond with a failure.

29. CashlessRevalue(AAA,BBB)

GUI command	
Command	**Parameters/Comments**
CashlessRevalue(AAA,BBB)	This command will recharge the customer's account, stored on payment media: - AAA – cashless number; - BBB – value to recharge. BBB should be <= than the LimitValue reported by the cashless device using command CashlessRevalueLimitRequest (see 28 for details)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"CashlessNumber": 1, "CashlessStatus": "RevalueApproved", "CashlessStatusCode": 13} or {"CashlessNumber": 1,"CashlessStatus": "RevalueDenied","CashlessStatusCode": 14} if the revalue is higher than the revalue limit reported by the cashless device or {"MDBCashlessRevalueLimitRequest": -1} – on failure communicating with cashless device.	

30. CashlessCashSale(AAA,BBB,CCC)

GUI command	
Command	**Parameters/Comments**
CashlessCashSale(AAA,BBB,CCC)	This command will report a cash only sale to the cashless device if this supports the option (see cashless settings command). This command is for statistic purposes, if the cashless device supports audit or sales reporting: - AAA – cashless number; - BBB – product price; - CCC – product number.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"MDBCashlessCashSale": 0} – on success or {"MDBCashlessCashSale": -1} – on faliure communicating with cashless device	

31. MDBSendRaw(A,B,C,D,…,N)

GUI command	
Command	**Parameters/Comments**
MDBSendRaw(A,B,C,D,…,N)	This command offers the possibility to the user's application to send any other desired command to the MDB bus. The last byte should be the CRC calculated by the MDB rule (see MDB manual for details). Bytes value could be sent in decimal format (0-255) or in hexadecimal format (0x00-0xFF), even mixed (decimal and hexadecimal in the same message)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
MDB response	The response will vary, depending on the sent command, according to the MDB manual. The interface will only convert 8 bit to 9 bit and back to 8 bit and will ACK/NACK in the 5ms interval required, according to the MDB specifications.

31. MDBSendRawCRC(A,B,C,D,…,N)

GUI command	
Command	**Parameters/Comments**
MDBSendRaw(A,B,C,D,…,N)	This command offers the possibility to the user's application to send any other desired command to the MDB bus. The users is sending the message bytes only, and the CRC will be calculated and added by the daemon. Bytes values could be sent in decimal format (0-255) or in hexadecimal format (0x00-0xFF), even mixed (decimal and hexadecimal in the same message)
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
MDB response	The response will vary, depending on the sent command, according to the MDB manual. The interface will only convert 8 bit to 9 bit and back to 8 bit and will ACK/NACK in the 5ms interval required, according to the MDB specifications.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. RTCSet

GUI command	
Command	**Parameters/Comments**
RTCSet(hh,mm,ss,dd,MM,yy,dow)	This command sets the internal RTC. Parameters are: - hh – hour (00→23) – device only supports 24h time format; - mm – minutes (00 → 59); - ss – seconds (00 → 59); - dd – day (00 →31); - MM – month (00 → 12); - yy – year (00 → 99); - dow – day of week (00 → 07 where 0 is for Sunday and 7 for Saturday) Example: RTCSet(11,51,00,21,3,18,3) This commands sets the internal RTC to 11:51:00, March 21st, 2018 on a Wednesday.
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{“RTCSet": 0} - success or {“RTCSet": -1} - failed	

38. RTCGet

GUI command	
Command	**Parameters/Comments**
RTCGet	
RASPIVEND daemon answer	
Answer	**Parameters/Comments**
{"RTCGet": [hh,mm,ss,dd,MM,yy,dow]}	This command gets the internal RTC timestamp - hh – hour (00→23) – device only supports 24h time format; - mm – minutes (00 → 59); - ss – seconds (00 → 59); - dd – day (00 →31); - MM – month (00 → 12); - yy – year (00 → 99); - dow – day of week (00 → 07 where 0 is for Sunday and 7 for Saturday) Example: {"RTCGet": [12,23,24,21,3,18,3]} This response means the time is 12:23:24 on March 21st, 2018 on a Wednesday

NOTES: