UBRIDGEEXE

User's manual

v1.0 - Executive (Protocol A) to TTL/RS232/USB

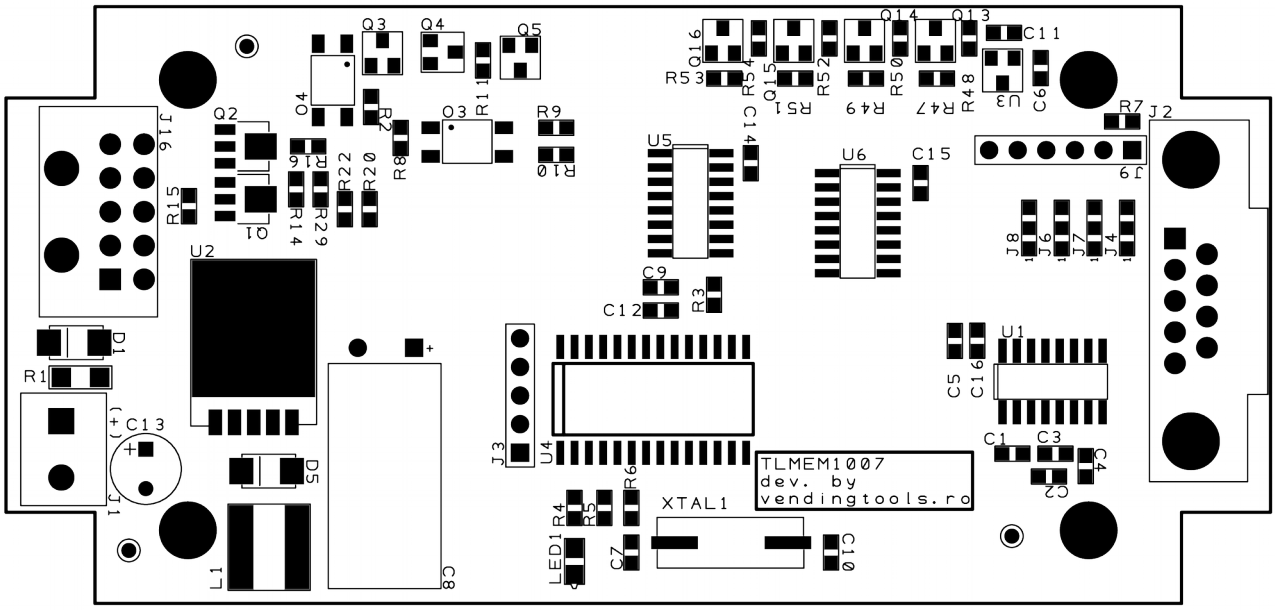
Table of Contents

I. General description

A. Overview

This is an universal module that can be used as a bridge between any TTL (3V3, USB or RS232 device or computer/PC/Raspberry Pi, etc.) and an Executive (Protocol A) enabled vending machine.
It offers a very simple serial protocol that can be easily implemented in any programming language, without the need of learning Executive protocol and managing some low level restrictions that are needed to connect to the vending machine.
Also, we are offering a Python 3 demo application that can be used at a wrapper or as a sample to speed up your development.

B. Hardware



Connector description:
- **J1 – Power connector**
    - PIN1 (squared pin) → VDC (8 to 36VDC)
    - PIN2 → GND

- **J2 – Serial port** - RS232
    - PIN1 (squared pin) – not used
    - PIN2 – serial TX
    - PIN3 – serial RX
    - PIN4 – not used
    - PIN5 – GND
    - PIN6 – not used
    - PIN7 – RTS (IN)
    - PIN8 – CTS (OUT)
    - PIN9 – not used

- J9 – Serial port – TTL 3V3
- PIN1 – 5VCC
- PIN2 – GND
- PIN3 – TX
- PIN4 – RX
- PIN5 – RTS (IN)
- PIN6 – CTS (OUT)

- J16 – Vending machine connector
- PIN1 – Power - VDC applied on J1
- PIN2 – Power – 5VDC
- PIN3 – GND
- PIN4 – Executive TX to VMC
- PIN5 – GND
- PIN6 – EXCUTIVE RX from VMCstat
- PIN7 – GND
- PIN8 – not used
- PIN9 – not used
- PIN10 – not used

On power-up, if the machine is “out of order” for various reasons (self check, water heating, etc.), the interface will wait for it to become available. In this interval, it will not answer to any external command being busy waiting the machine. If the machine is not becoming available in 4 minutes, the interface will abort it’s monitoring and will enter in idle mode. In this mode it will answer to any command, and the getstatus command will return 0x40 to the VMC status, meaning that the machine is out of order (see GetStatus message below).

II. Low level communication protocol

A. Serial port settings

Low level communication protocol is fairly simple. To use this protocol you need to be able to directly use to the serial port where the device is connected.
Communication parameters for serial port are:

- Speed – 115200bps;
- Data bits – 8;
- Stop bits – 1;
- Parity – none;
- Flow control – hardware – RTS/CTS.

The low level protocol is consisting in a set of binary messages that can be exchanged between the host (PC) and the interface.
For the rest of the document, we will use the following conventions:
- B_n = byte number “n”
- b_n = bit number “n”
- VMC = represents vending machine controller (the vending machine)
- PMS = represents payment systems

Each message has the following structure:

CMD	SUBCMD	CONTENT	CRC
Message header	Sub-command	Message content	XOR from CMD to the last DATA byte
Example: sending a credit of 100 with available change flag			
0xFA	0x05	0x00 0x00 0x00 0x64 0x01	0x9A

There are 2 special messages that are not respecting the above structure:
- ACK 0xFC 0xFC 0xFC 0xFC 0x00
- NACK 0xFD 0xFD 0xFD 0xFD 0x00
For messages that are not requiring other response, an ACK or NACK should be issued as a response. Also ACK or NACK is required for unsolicited messages received from the device. If the device will not receive ACK to an unsolicited message within 1 second or if it will receive NACK, it will retry to resend the last message for 3 times. If it cannot receive a successful ACK, the device will abort sending current message and it will resume to the next task.

B. Device memory map

The device is storing some informations into it's non-volatile memory. Memory can be read and written to modify device's setting, to read sales counters and to reset those counter when needed. Below you can find the device memory map:

Address (decimal)	Length (bytes)	Content description
0 → 383	384	Prices. Starting with address 0, there are stored 96 prices, each price with a 32bit value. For example, the price for product (selection) number 2 is stored at address 4 as follows: - address 4 – B_3 (MSB) - address 5 – B_2 - address 6 – B_1 - address 7 – B_0 (LSB) This counter may be read or write using read/write commands described below. For example, can be set to 0 after the filler's visit
384	1	Vending setting byte. Those are few flags needed by the interface to manage some vend settings: - b_0 – not used in this version - b_1 – if this bit is set to 1, then the interface is working on Executive "price holding" mode (prices are set on the interface). If this bit is cleared to 0, then the interface is working on Executive with prices held by the machine. - b_2 – not used in this version - b_3 – not used in this version - b_4 – display/don't display price – if this bit is set to 1, then the interface will immediately send price as credit to the machine when a product is selected, but the credit is 0. Depending on the machine, this behavior will force the machine to display the product's price if the product is selected and the credit is 0. Not all the machines are correctly responding to this instruction. If you fine some strange behaviors on your machine (display huge or wrong credit when selecting a product while credit is 0) then you must disable this feature by clearing this bit to 0. - $b_5 \rightarrow b_7$ – not used in this version After modifying this setting, the interface must be initialized by a reset command to give it the opportunity to reload it's settings.
385	1	Scaling factor – this byte sets the interface's scaling factor. It has to match the settings on the vending machine, otherwise strange behaviors may occur on credit display. After modifying this setting, the interface must be initialized by a reset command to give it the opportunity to reload it's settings.
386	1	Decimal places – this byte sets the interface's decimal places. It has to match the settings on the vending machine, otherwise strange behaviors may occur on credit display. After modifying this setting, the interface must be initialized by a reset command to give it the opportunity to reload it's settings.
831 → 1023	192	Sales counters. Starting with address 831 there are stored 96 counters with 16bit value. For example, the counter for product (selection) number 2 can be found at address 8331, as follows: - address 833 – counter for product number 2 (most significant byte) - address 834 – counter for product number 2 (less significant byte) This counter may be read or write using read/write commands described below. For example, can be set to 0 after the filler's visit.

C. Low level protocol description

1. Write to memory

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x01	- B_2 – HI byte of address to write - B_3 – LO byte of address to write - B_4 – B_n – data bytes to be written (maximum 64byte for one write operation)	CRC
Device response			
ACK or NACK			

2. Read from memory

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x02	- B_2 – address to start reading (MSB) - B_3 – address to start reading (LSB) - B_4 – number of bytes to be read (MSB) - B_5 – number of bytes to be read (LSB)	CRC
Device response			
0xFA	0xF2	- B_2 – address to start reading (MSB) - B_3 – address to start reading (LSB) - B_4 – number of bytes to be read (MSB) - B_5 – number of bytes to be read (LSB) - B_6 – B_n – data bytes to be read (maximum 64 for one read operation)	CRC

3. Reset the device

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x03	NONE	CRC
Device response			
ACK or NACK			

4. Get status

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x04	NONE	CRC
Device response			
0xFA	0x04	- $B_2 \rightarrow B_9$ – device serial number - B_{10} – current credit (MSB) - B_{11} – current credit - B_{12} – current credit - B_{13} – current credit (LSB) - B_{14} – VMC status (oldest status) - B_{15} – VMC status - B_{16} – VMC status - B_{17} – VMC status (most recent status) If the most recent status is 0x40, the machine is out of order and if it is 0x00, the machine is up and running.	CRC

5. Send credit to VMC

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x05	- B_2 – credit (MSB) - B_3 – credit - B_4 – credit - B_5 – credit (LSB) - B_6 – change mode (if 0x01 – there is available change, if 0x00 – there is no available change – either not enough coins in the changer or the machine does not have a coin changer – the machine will display “Use exact change” or “No change” or some similar message	CRC
Device response			
ACK or NACK			

6. Read current credit

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
0xFA	0x06	NONE	CRC
Device response			
0xFA	0x06	- B_2 – current credit (MSB) - B_3 – current credit - B_4 – current credit - B_5 – current credit (LSB)	CRC

7. Machine’s status changed – unsolicited message

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
NONE	NONE	NONE	NONE
Device response			
0xFA	0x07	- B_2 – if 0x01 – machine changed it’s status in “available” (working and idle), if 0x00 – machine changed it’s status in “out of order” (some problem occurred and cannot sell anymore)	CRC

8. Machine starts delivering a product – unsolicited message

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
NONE	NONE	NONE	NONE
Device response			
0xFA	0x08	NONE	CRC

This message comes from the machine every time the interface is sending the “vend” approval to the vending machine. After issuing this message, the device will not respond to any command until the machine finishes the delivery (success of fail) because it is monitoring the machine’s response. If the machine is not sending any response within 2 minutes, then the interface will resume on the nest task and will consider the vend was a failure, not taking any amount from the available credit. The end of the vend activity will be signaled by the device sending the “vend result” message (see below “Vend result – unsolicited message

9. Vend request – unsolicited message

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
NONE	NONE	NONE	NONE
Device response			
0xFA	0x09	- B_2 – selection (product) number - B_3 – price (MSB) - B_4 – price - B_5 – price - B_6 – price (LSB)	CRC

10. Vend result – unsolicited message

CMD (B0)	SUBCMD (B1)	CONTENT	CRC (B_{n+1})
NONE	NONE	NONE	NONE
Device response			
0xFA	0x0A	- B_2 – vend result – if 0x01 – vend was successfully close, if 0x00 – vend failed for some reasons.	CRC

III. High level communication protocol

This protocol is facilitated by a Python 3 wrapper that listens on a socket for simple commands translates them to low level protocol and send them to the serial port, also receiving and parsing device's messages and translate them to JSON messages that will be sent back to the socket. In this manner you can use telnet or similar application to manipulate credit, sales, reading and writing settings and counters, etc.
The Python 3 wrapper depends on PySerial 3.0.1 which must be previously installed.
For read and write to memory, please see the device's memory map on page 6.

1. Write to memory

Message to the device	Explanation
writemem(address,$data_1$,$data_2$...$data_n$)	- <address> is a decimal value between 0 and 1023 – please see memory map on page 6 for more details - <data> are bytes, separated by comma (,) - $data_1$ is written and <address>, <$data_2$> is written at <address> + 1 and so on. Maximum number of bytes that can be written on a single write operation is 64. Example: writemem(800,1,2,3,4,5,6,7,8,9,0) will write, starting to address 800, values 1 ad address 800, 2 at address 801 and so on
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

2. Read from memory

Message to the device	Explanation
readmem(address,data_length)	- <address> is a decimal value between 0 and 1023 – please see memory map on page 6 for more details - <data_length> is the number of bytes to be read. The maximum number of bytes that can be read on a single command is 64. Example: readmem(800,10) – reading 10 byte starting with address 800
Response from the device	**Explanation**
"{DeviceMessage":"ReadMemory", "StartAddress":"800", "DataLength":"10", "MemoryData":[\x01,\x02,\x03,\x04,\x05,\x06,\x07,\x08,\x09,\x00]}	It returns a JSON with an array of bytes containing the content of the memory starting with the requested address and with the requested length

3. Reset

Message to the device	Explanation
reset	This command will reset the interface. It is required after modifying vending settings. During the reset procedure, the interface will not respond to any other command (for about 4 seconds from the correct ACK message received from the device).
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

4. Get interface's status

Message to the device	Explanation
getstatus	This command will ask the interface's status
Response from the device	**Explanation**
{DeviceMessage":"DeviceStatus", "DeviceSerialNumber":"10000011", "CurrentCredit":"0", "VMCStatus":[\x00,\x00,\x00,\x00]}	The VMC status in this message represents the latest 4 status changes of the vending machine. If the last value is 0x40, then the machine is out of order. If the last status is 0x00, then the machine is running and in idle state.

5. Send credit to the machine

Message to the device	Explanation
sendcredit(value,change_mode)	- <value> is the credit value to send on the machine - <change_mode> is the current change status – 1 → change is available – 0 → change is not available (machine will show a message like “Machines returns no change”, “Exact change”, etc. Example: sendcredit(1000,1) will send EUR10.00 to the machine if the decimal places are set to 2 and will signal to the machine that is enough credit available.
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

6. Read price

Message to the device	Explanation
readprice(product_number)	This command will ask the price for <product_number> Example: readprice(5)
Response from the device	**Explanation**
{"DeviceMessage":"ReadPrice", "ProductNumber":"5", "ProductPrice":"500"}	

7. Write price

Message to the device	Explanation
writeprice(product_number,product_price)	This command will set the price for <product_number> Example: writeprice(3,150) will set the price at EUR1.50 for product number 3.
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

8. Read scaling factor

Message to the device	Explanation
readscalingfctor	This command will read the saved scaling factor stored on the device Example: readscalingfactor
Response from the device	**Explanation**
{"DeviceMessage":"ReadScalingFactor", "ScalingFactorValue":"10"}	

10. Write scaling factor

Message to the device	Explanation
writescalingfactor(n)	This command will set the scaling factor on the device. Example: writescalingfactor(10) will set the device scaling factor to 10. This command needs a device reset by issuing reset command, to become effective
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

11. Read decimal places

Message to the device	Explanation
readsdecimalplaces	This command will read the saved decimal places setting stored on the device Example: readdecimalplaces
Response from the device	**Explanation**
{DeviceMessage":"DeviceStatus", "DeviceSerialNumber":"10000011", "CurrentCredit":"0", "VMCStatus":[\x00,\x00,\x00,\x00]}	

12. Write decimal places

Message to the device	Explanation
writedecimalplaces(n)	This command will set the decimal places on the device. Example: writedecimalplaces(2) will set the device decimal places to 2. This command needs a device reset by issuing reset command, to become effective
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

13. Read product counter

Message to the device	Explanation
readproductcounter(n)	This command will read the counter for product “n”, stored in the device’s memory. For each sale, the corresponding counter will be updated, so your application can read counters for stock reporting and alerts. Example: readproductcounter(1)
Response from the device	**Explanation**
{"DeviceMessage":"ReadProductCounter", "ProductNumber":"1", "ProductSalesNumber":"14"}	

14. Write product counter

Message to the device	Explanation
Writeproductcounter(n,x)	This command will set the “n” product counter to value “x”. This command can be useful to reset the counter on filler’s visit or tu adjust it depending on the situation. Example: writeproductcounter(1,0) – will reset the counter for product number 1.
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

15. Read vending settings

Message to the device	Explanation
readvendingsettings	This command is used to vending settings byte from device’s memory. Please see address 384 in memory map table (page 6) for details.
Response from the device	**Explanation**
{"DeviceMessage":"ReadVendingSettings", "VendingSettingsValue":"15"}	

16. Write vending settings

Message to the device	Explanation
writevendingsettings(n)	This command will set the vending settings on the device. Example: writevendingsetings(2) will set the device to not display prices and work in price holding mode. Please see address 384 in memory map table (page 6) for details. This command needs a device reset by issuing reset command, to become effective
Response from the device	**Explanation**
{"DeviceMessage":"Acknowledge"} {"DeviceMessage":"NegativeAcknowledge"}	

17. Read current credit

Message to the device	Explanation
readcurrentcredit	This command is used to read the current credit on the device. This is the same with the credit displayed on the vending machine. Useful when a transaction ends and the customer requires the change. Example: readcurrentcredit
Response from the device	**Explanation**
{"DeviceMessage":"ReadCurrentCredit", "CurrentCreditValue":"350"}	This response means that the current credit on the machine is EUR3.50

18. Read serial number

Message to the device	Explanation
readserialnumber	This command is used to read device's internal serial number Example: readserialnumber
Response from the device	**Explanation**
{"DeviceMessage":"ReadSerialNumber", "SerialNumber":"10000011"}	

19. Machine's status changed – unsolicited message

Message to the device	Explanation
NONE	
Response from the device	**Explanation**
{"DeviceMessage":"MachineStatus","Status":"Disabled"} {"DeviceMessage":"MachineStatus","Status":"Enabled"}	This message will be sent by the device when the machine changes it's status from "enabled" to "disabled" or from "disabled" to "enabled" - "enabled" means the machine is running and idle and waiting for credit - "disabled" means the machine is out of order for some reasons and is not able to receive credit.

20. Machine vend request – unsolicited message

Message to the device	Explanation
NONE	
Response from the device	**Explanation**
{"DeviceMessage":"VendRequest", "ProductID":"2", "ProductPrice":"150"}	This message will be send by the device when the customer has made it's selection. If there is enough credit previously sent to the device, it will send the permission to sale. In the example on left, the customer selected the product number 2 with a price of EUR1.50. This function can be used for real time reporting purposes.

21. Machine vend started – unsolicited message

Message to the device	Explanation
NONE	
Response from the device	**Explanation**
{"DeviceMessage":"VendStatus", "Status":"VendStarted"}	This message will be send by the device once the machine is starting the delivery process. During the delivery process, the device is not responding to any external command on the serial port since it is busy monitoring the vending machine. If the machine is not reporting the delivery process in 2 minutes, then the device will abort monitoring and consider that the transaction failed. After the 2 minutes timeout, the device will start responding to commands on serial port

22. Machine vend result – unsolicited message

Message to the device	Explanation
NONE	
Response from the device	**Explanation**
{"DeviceMessage":"VendResult","ResultCode":"VendSuccess"} {"DeviceMessage":"VendResult","ResultCode":"VendFailed"}	This message will be send when the machine finished the delivery and reported to the device the delivery status. If "VendSuccess", then the device will get the product price from the total current credit and will display on the machine the remaining credit. If "VendFailed", then the device will keep the initial credit. This command ends the device's non-responding period initiated by the "vend started" message. Right after sending this message to the host it will start responding again to the commands on the serial port.