Velbus Starter Guide

written by Golfy
2012 January, 25th

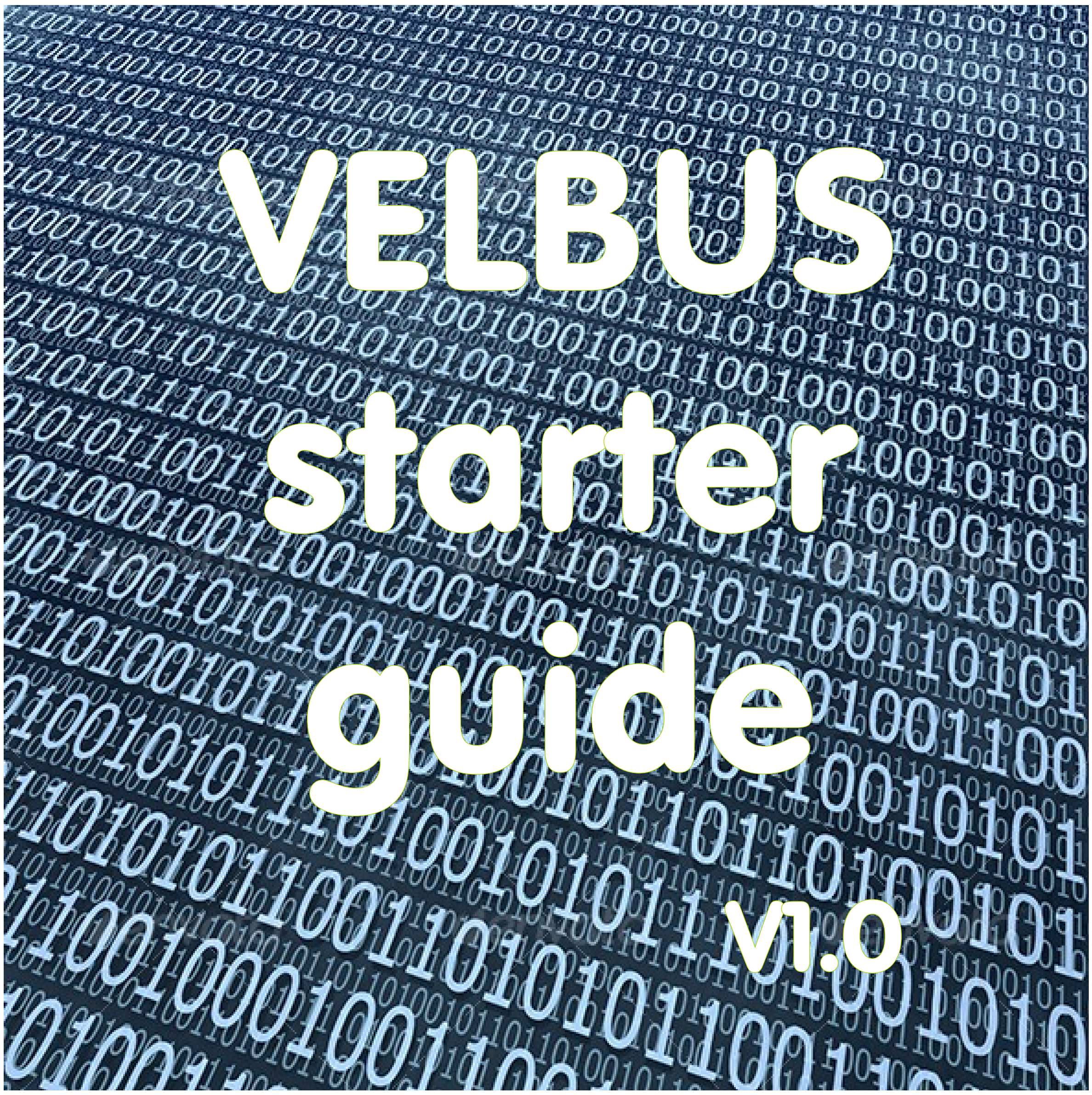


Table des matières

Introduction

This small document is just a kind « Velbus for dummies » : it would introduce notions but if you want to learn more, you'll have to experiment, to ask for questions and to search on Internet.

As I'm a bad programmer and there are a lot of language, this document doesn't show any example.

As I'm french, my english is a little rusty (but it's worse in oral ;)) : it my sentences aren't clear, you can inform me on the Velbus forum or by email.

Take it like an help to use official Velbus protocol specification.

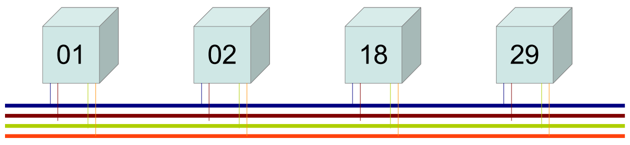
If you think something is wrong, please contact me : I'll correct it as soon as possible : mailto:golfy@free.fr

Thanks,

David

Physical BUS and messages

As each module are attached to on the same wires, the communication should be serial : only one module can speaks at a time. Each module have it own address, like an unik identifier.



A Bus is physicaly constitued by 4 wires :

- Power + wire
- Power – wire
- Communication + wire
- Communication – wire

Power wires are just connected to give energy to each module : you can also connect a lamp on it because it's 12 – 24v power DC. However, it's not recommanded, only for testing if power is present. The better way is to use a multimeter (powermeter).

Communication wire are more complex to understand : first, it's not a DC signal but not AC signal too ! In fact, the only way to know what is on these wires is to use an oscilloscope. It would probably shows a strange squared signal, not cyclic nor always the same. It's a numerical signal.

Each bit are transmit from module to module by these wires : 8 bits is a byte and Velbus BUS use messages that are concatenation of a lot of bytes.

A multimeter isn't able to analyse these communication wires, so it's important to use always the same colors convention (L and H means Low and High : you can use hot colors for High and cold colors for Low).

Velbus configuration

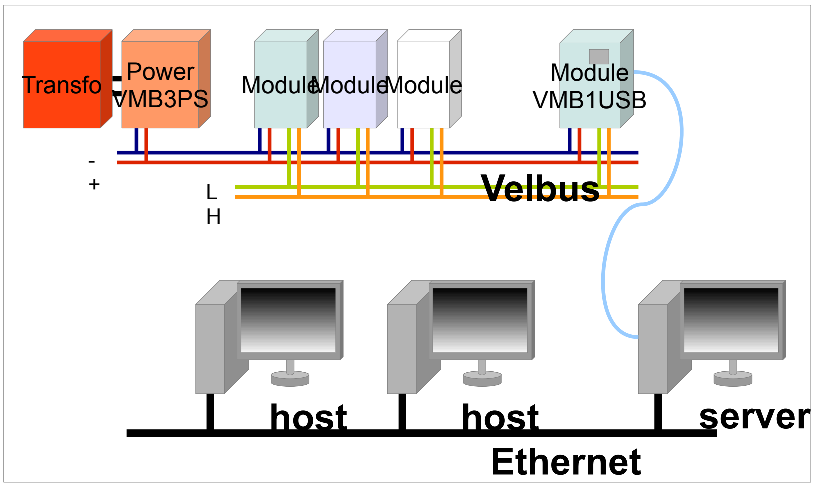
Basic configuration

In order to play with Velbus BUS with a computer, you need to use these modules :

- Transfomer VMB1 (230v – 21v)
- Power supply module (regulator 3x1A)
- Modules (at least a VMB4RY or VMB1BL...)
- Communication module (VMB1USB or VMB1RS)

Full configuration

This architecture is a full one because there is a server (it's a simple PC with an application which can translate message from Velbus BUS to Ethernet BUS). The advantage is that you can connect with any other PC to Velbus



How it works ? The server application start the connexion to Velbus trhough VMB1USB (or VMB1RS for old PC). This application needs to know the COM port and on wich Ethernet port it has to wait client.

When a host would to connect to Velbus (with VelbusLink for example), it has to connect to the server : then the server application can copy datas from Velbus to Ethernet et from Ethernet to Velbus.

You can find many server application on the Velbus forum :

Velbus-Server mode tools

Logical BUS and messages

Hardware components are present on each module and manage how to use the physical BUS :

- Speed
- Parity
- Collision
- Priority

Velbus BUS is inspired by CAN BUS : they're used in industrial equipements and automotive.

However, each module must be able to know if a message is destinated to him or nor. A protocol should describe how message are delimited : when it start and when it ended.
Velbus gives some documentation about it but here is a resume :

0F	**##**	**##**	**##**	**##**	**04**
Start byte				Control byte	End byte

Bytes are noted in hexadecimal mode : 0F is equal to 15 in decimal. To avoid errors between hexadecimal and decimal mode, a prefix character is added for « human read ».
That means $0F is the hexadecimal value for decimal number 15.

In fact, hexadecimal is more easy to use on a computer (because computers counts in binary) : then character for hexadecimal number are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

so when 10 is coming after 9 in decimal mode, 10 is coming after F in hexadecimal. From this point, notation for hexadecimal numbers would be prefixed with '$' in text but not in BUS message. Sometime, you'll see the prefix '&H' : it depend of programming language. In this document, I'll use '$'.

For binary value, I'll use '%' : %00011111 mean $1F and value is 31 in decimal

Coming back to Velbus messages :

0F ## ## ## ## 04	0F ## ## ## 04	0F ## ## ## ## ## ## ## ## ## 04	...	0F ## ## ## ## 04

Delimiters are $0F byte and $04 byte : that's like an envelops ! We've to write information in it AND to write a kind of signature that prove the information has been correctly transmitted, like a stamp. This is the last byte before delimitation. It's a checksum byte

Checksum is calculated just before sending a message. It's a counter that sums all bytes before him.
Formule is :

$$\text{CHECKSUM} = (\sim\text{SUM})+1$$

That's a one byte sum with two complement.

Of course, messages could have the same length but in Velbus BUS protocol, it's not the case ! Let's zoom on the basic of the frame :

0F	**##**	**##**	**##**	**##**	**##**	**##**	**04**
Start byte	Priority : FB or F8	Address : 00=multicast	Data length / RTR	DATA	DATA	Control byte	End byte

As you can see, the length of the minimal frame is 6 bytes (with 0 byte of DATA).
There is a fixed part to read : the first 4 bytes.

However, messages are store in a small memory called buffer : if you try to read them on serial interface (COM port for SubD-9 or USB) or through network interface (Ethernet-Velbus server), this buffer will be flushed in your buffer's program. You've to be sure that the message is full (no part missing).

Note :
as messages are store in buffer memory, it's important to count correctly !

0F	**##**	**##**	**##**	**##**	**##**	**##**	**04**
Byte 1 Position 0	Byte 2 Position 1	Byte 3 Position 2	Byte 4 Position 3	Byte 5 Position 4	Byte 6 Position 5	Byte 7 Position 6	Byte 8 Position 7

Reading byte 3 means checking with byte is present in buffer+2 bytes

Functions and messages

Now, we need to understand how modules are communicating : they are two main functions.

Sensors

These modules are waiting for a contact and send a command message.

Sensors are VMB4PD, VMB6IN, VMB8PB...

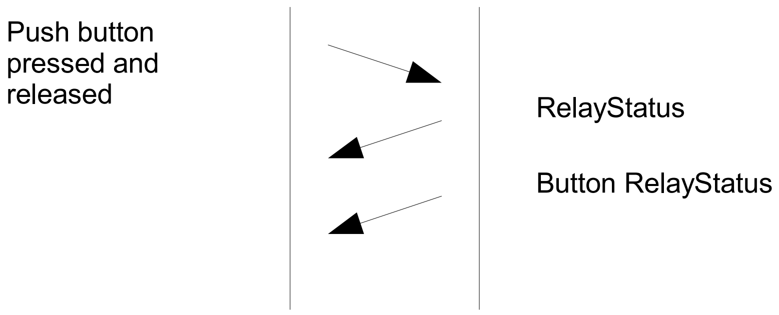
Actuators

These modules are able to interact with electrical components.

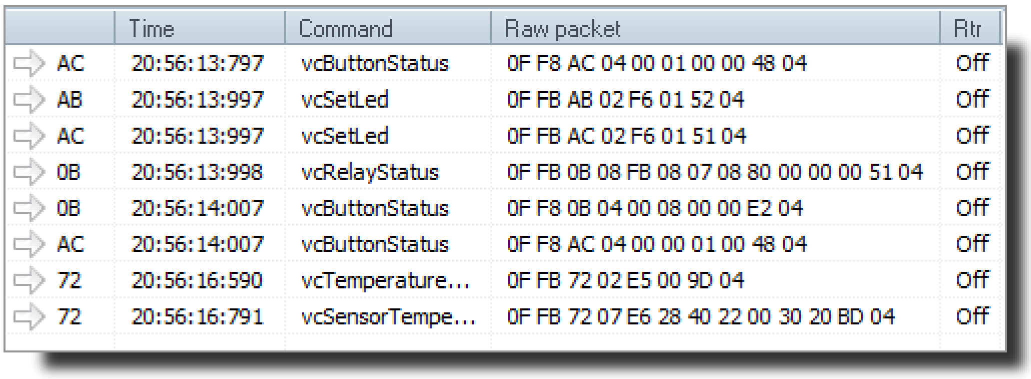
Actuators are VMB4RY, VMB1BL, VMBDM1...

The VelbusLink software create relations between sensors and actuators : clicking on a push button of a VMB4PD send a command message on the BUS. Each modules receive it but only those who were programmed to accept this command take the information. By feed-back, actuators could send command message for LED notification so message are often bi-directional.

So, communication could be see like this :



and it is possible to see these exchange with VelbuLink :



	Time	Command	Raw packet	Rtr
AC	20:56:13:797	vcButtonStatus	0F F8 AC 04 00 01 00 00 48 04	Off
AB	20:56:13:997	vcSetLed	0F FB AB 02 F6 01 52 04	Off
AC	20:56:13:997	vcSetLed	0F FB AC 02 F6 01 51 04	Off
0B	20:56:13:998	vcRelayStatus	0F FB 0B 08 FB 08 07 08 80 00 00 00 51 04	Off
0B	20:56:14:007	vcButtonStatus	0F F8 0B 04 00 08 00 00 E2 04	Off
AC	20:56:14:007	vcButtonStatus	0F F8 AC 04 00 00 01 00 48 04	Off
72	20:56:16:590	vcTemperature...	0F FB 72 02 E5 00 9D 04	Off
72	20:56:16:791	vcSensorTempe...	0F FB 72 07 E6 28 40 22 00 30 20 BD 04	Off

Delays between messages are very short (less than 1/10e second) but as you can see a simple push can introduce a lot of message (module with address $72 isn't concerned).
In this example, there are two sensors that are listen for the $0B relay's module : $AC (I've pressed it push button) and $AB (that's VMB8PB wich can control the same relay module).

An important point for BUS is addresses : like a street, each house have an unique number. This way, each module could be identified, programmed and can sent datas on the BUS. Be careful with these addresses because some of them are reserved by Velbus.

Common messages (all modules)

Now, you're able to understand how Velbus BUS works : it's like knowing that a human language use 'subject' and 'verb' and some other attributes.

Fortunely, Velbus has created a protocol with enough simplicity and now we'll learn it vocaburary :)

Clock Synchronisation

This first message is a **broadcast** one : destination isn't one or few modules but ALL modules, that's why the reserved address $00 is used.

0F	FB	00	04	D8	JJ	HH	MM	##	04
Start byte	Priority : FB (low)	Address : 00=multicast	No RTR 4 bytes length	Function Time Sync	Day number (0-6)	Hour (00-23)	Minut (00-59)	Control byte	End byte

Be worry about conversion :
Value for hour is in décimal but in VelbusLink, RAW data are written in hexadecimal.
23:48 will show $17 and $30 but effective values are 23 and 48

This frame is able to synchronize all modules who are using timer : VMB4PD, VMB8PBU, VMB8PBN...

It's the most simple message as there is no answer : it's a PUSH message.