## Explanations about the Vector CAN database

***Overview***

The Vector CAN database (*.dbc), that should come together with this documentation, is intended to be used to implement power supplies or electronic loads into Vector company software like CANalyzer or CANoe. This database defines the most important commands available for a particular device, plus necessary calculation factors.

***Limitations***

Due to the structure of the database, not all available commands can be addressed to signals. However, the documentation of the interface hardware (here: CAN interface IF-C1) includes so-called object lists that list all available commands for a certain device series. The remaining commands can be implemented into the database and a related configuration by the use of CAPL, but also require extra handling.

***What's necessary?***

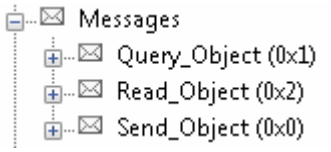
A device with CAN interface card IF-C1 or IF-C2, a dedicated database (for example PS8080-510_3U.dbc), Vector software or compatible software, CAN controller hardware.

***What setup is required on the device side?***

The user has to adjust the CAN base ID (see device and/or interface card documenation) on the device, along with other typical CAN communication settings. When using multiple units on the same bus, the CAN base ID of every unit has to be different.

***What setup is required on the PC side?***

The user has to open the database editor CANdb++ in order to adjust the CAN IDs of the basic messages so they match the device IDs. Every device gets three CAN IDs, based on the adjustable CAN base ID. The default IDs in the database are 0, 1 and 2.




It applies:
**Query_Object** = Base ID + 1
**Read_Object** = Base ID + 2
**Send_Object** = Base ID,
where the base ID is the CAN base ID as adjusted on the device.
Example: the device has been set to base ID 500. Then the three messages in the database have to be assigned the IDs: **Query_Object (501), Send_Object (500), Read_Object (502)**

***How to use the three messages?***

These three messages handle the complete communication with the device.

**Send_Object:** Send set values (U, I, P etc.) and device condition (output on/off etc.)
**Query_Object**: Query set values, actual values and status (CV etc.)
**Read_Object:** This is used to read data into signals, queried from the device with Query_Object.

Use of **Send_Object**
Send_Object is used to send set values or device conditions to the device. It is required to a) select which set value/device condition to set and b) to give the set value/device condition itself. The selection of the set value or device condition is done via signal Send_Mux, which is connected to a symbolic table that enables to select like from „Output voltage".
Set values are given as real values, e.g. 40V as 40. Conditions like „Output on" have to be selected in two parts (two bytes), because multiple conditions can be set with one command.
Example: in order to switch the DC output of a power supply on, for **Send_Object** the signal Send_Mux (0x36) has to be selected first, then signal Send_Control_Mask „Output" and then signal Send_Output „on".

Use of **Query_Object**
This message is used to query information from the device and thus just required to tell what information is queried. Selection is done with signal Query_Mux and the related, symbolic table which is self-explaining.

Use of **Read_Object**
This message automatically sorts incoming information based on the ID into signals. Usually, every command that is used with **Query_Object**, will cause the device to return data on the base ID +2, that should be assigned to **Read_Object**. The data in the signals can be used for further purposes.

***How to interprete the incoming data from the device?***

- Set values and actual values are calculated into real values by the CAN software
- Statuses like „Output on/off“ are described in the assigned value tables
- Alarm codes and types like „OT“ are described in the assigned value tables

***Notes***

- In case a value is read from a device and displayed in a typical Vector panel output box and that box is marked in red while the value is actually not exceeded, this is caused by calculation errors. In order to avoid that you just need to lower the last digit of the corresponding calculation factor by 1.