Implementation of Siemens USS Protocol into LabVIEW

P. Hosek[1], M. Diblik[2]

[1] Technical University of Liberec, Faculty of Mechatronics, Institute of New Technologies and Applied Informatics, Studentská 2, 461 17 Liberec 1, Czech Republic
[2] Technical University of Liberec, Faculty of Mechatronics, Institute of Mechatronics and Computer Engineering, Studentská 2, 461 17 Liberec 1, Czech Republic

Introduction

As a part of the apparatus for the valve train measurements on Škoda 1.2HTP combustion engine we utilized 2-pole, 7.5kW asynchronous electromotor, frequency inverter SINAMICS G120 and control unit CU240S, all from Siemens AG. The electromotor is used to run the crank shaft of the combustion engine. The inverter, control unit and quadrature encoder assure the vector control of the electromotor. The transition of the torque between the electromotor and the crank shaft ensures ribbed belt.

The measurement of the kinematic values of the valve train is performed on predefined spectrum of speeds. For the control and monitoring of the speed it was necessary to create application for communication with the SINAMICS inverter. During the design of the measurement apparatus we also had to decide what communication bus and protocol to use. Since we develop a computer based superior system that ensures communication, data acquisition, data processing and failure handling to perform fully automated measurements and analysis of kinematics of the valve train, it was necessary to choose such an interface that would offer simple but reliable communication between the electromotor and the PC. Siemens offers wide spectrum of connection possibilities of their control units that operate the SINAMICS G120 inverter. The control units are divided:

Standard control units (without fail-safe functions):

- CU240S standard version of the CU240 control units – USS protocol (RS485)
- CU240S DP - PROFIBUS DP (RS485)
- CU240S PN – PROFINET (Ethernet)
- CU240E economic version of the CU240 control units (e.g. less terminals, no encoder interface)

Control units with fail-safe functions:

- CU240S DP-F like CU240S DP plus integrated fail-safe functions
- CU240S PN-F like CU240S PN plus integrated fail-safe functions

Since we have just simple topology PC(master)-Inverter(Slave) we have chosen the standard version of the control unit that allows us to connect over RS485 and communicates via the USS protocol.

We develop the measurement and control application in LabVIEW development environment so the essential question became the implementation of the USS protocol since no LabVIEW command libraries are available. There is of course the option to use the OPC server either from Siemens or 3rd party but it requires extra money, installation and setting of another application and driver not mentioning always not the best performance and the necessity to resolve compatibility issues. There was also permanent demand from the community of LabVIEW users to have native LabVIEW implementation of the USS protocol. Based on above written we decided to write collection of the most common USS commands. The final library was published for public use on National Instruments discussion forum – Motion Control and Motor Drives board.

Basic protocol overview

The USS protocol (Universal Serial Interface Protocol) defines an access technique according to the master-slave principle for communications via a serial bus. This also includes the point-to-point connection.

Essential features of the USS protocol are:

- It supports a multi-point-capable coupling, e.g. EIA RS 485 hardware
- Master-slave access technique
- Single master system
- Max. 32 nodes (max. 31 slaves)
- Simple, reliable telegram frames
- Easy to implement
- Operation with either variable or fixed telegram lengths

One master and a maximum of 31 slaves can be connected to the bus. The individual slaves are selected by the master via an address character in the telegram. A slave itself can never transmit without first being requested to do so, and direct message transfer between the individual slaves is not possible. Communications is realized in the half-duplex mode. Each transferred character starts with a start bit and ends with a stop bit. 8 data bits are transferred. Each character (byte) has a parity bit (even parity, i. e. the number of logical ones in the data bits, including the parity bit, is an even number).

Structure of USS telegram

Description

The following figure shows the structure of a typical USS telegram.

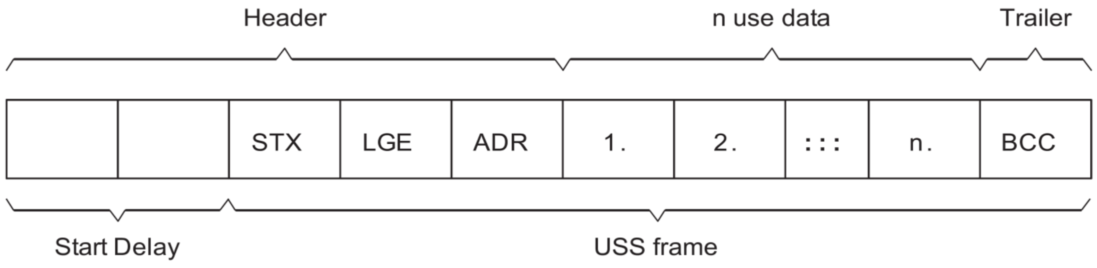


Fig. 1 – Structure of USS telegram [1]

Variable length telegrams and fixed length telegrams can both be used. This can be selected using parameters P2012 and P2013 to define the PZD and PKW lengths. The fixed length sizes we used are shown below:

STX	1 byte	
LGE	1 byte	
ADR	1 byte	
Use data	PKW	4 words
	PZD	8 words
BCC	1 byte	

SUM	28 bytes	

STX

The STX field is a single byte ASCII STX character (0x02) used to indicate the start of a message.

LGE

The LGE is a single byte field, indicating the number of bytes which follow this in the telegram. It is defined as the sum of:

- use data characters (quantity n=24 in our case)
- address byte (ADR)
- block check character (BCC)

The actual total telegram length will of course be two bytes longer as STX and LGE itself are not counted in the LGE.

ADR

The ADR field is a single byte containing the address of the slave node (e.g. inverter). The individual bits in the address byte are addressed as follows:

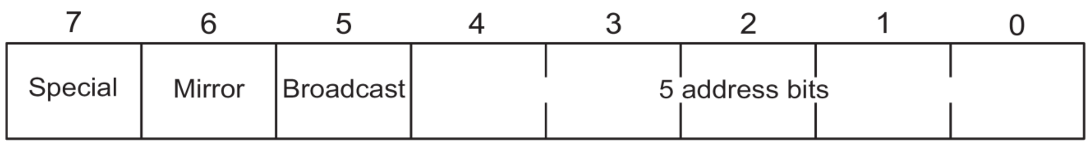


Fig. 2 – Structure of ADR byte [1]

- Bit 5 is the broadcast bit. If it is set to 1, the message is a broadcast message and will be acted upon by all inverters on the serial link. The node address is not evaluated. The USS protocol specification requires certain settings in the PKW area, refer to the later example on using USS broadcast mode.
- Bit 6 = 1 indicates a mirror telegram. The node address is evaluated and the addressed slave returns the telegram unchanged to the master.

- Bit 5 = 0 and bit 6 = 0 and bit 7 = 0 means normal data exchange for units. The node address (bit 0 … bit 4) is evaluated.

BCC

BCC means block check character. It is an exclusive OR (XOR) checksum over all telegram bytes except the BCC itself.

Structure of use data block

The use data area of the USS protocol is used for transferring the application data, which are the parameter channel PKW and process data channel PZD as can be seen on the following picture.

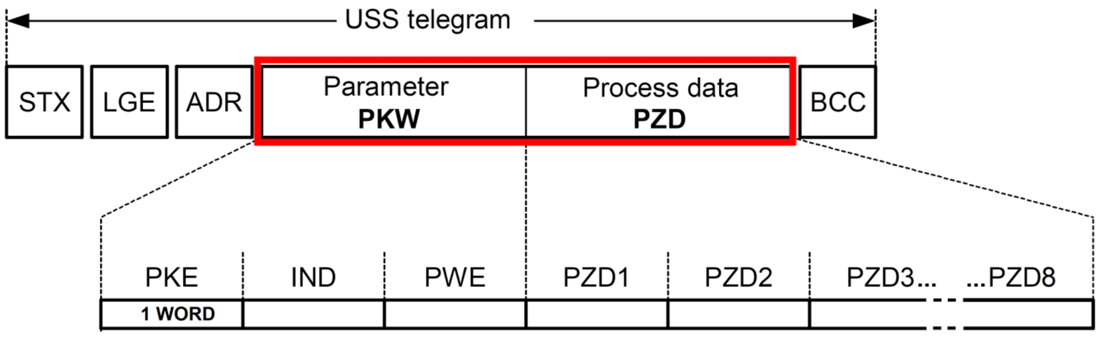


Fig. 3 – Structure of use data block [2]

Parameter channel PKW

The parameter channel can be used to monitor and/or change any parameter in the inverter.

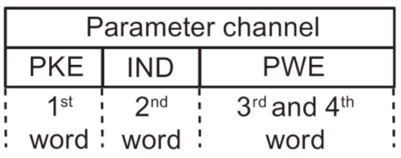


Fig. 4 – Structure of parameter channel [1]

Parameter identifier PKE, 1st word

PKE is always a 16-bit value and consist of following structure:

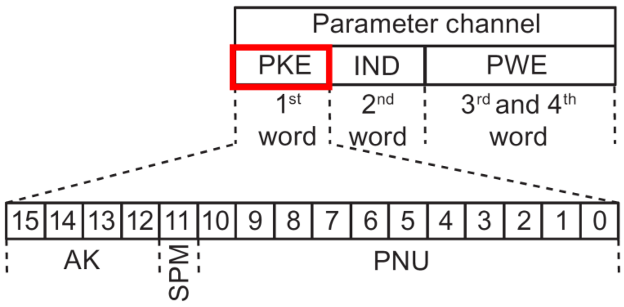


Fig. 5 – Structure of parameter identifier [1]

- Bits 0 to 10 (PNU) contain the remainder of the parameter number. For parameter numbers ≥ 2000 it is necessary to add an offset which is defined with the Page index of IND.
- Bit 11 (SPM) is reserved and always = 0.
- Bits 12 to 15 (AK) contain the request or the response identifier. The meaning of the request identifier for request telegrams (master → inverter) is shown in the table below.

Table 1 - Request identifier (master → inverter) [1]

Request identifier	Description
0	No request
1	Request parameter value
2	Modify parameter value (word)
3	Modify parameter value (double word)
4	Request descriptive element $^{1)}$
6	Request parameter value (array) $^{1)}$
7	Modify parameter value (array, word) $^{1)}$
8	Modify parameter value (array, double word) $^{1)}$
9	Request number of array elements
11	Modify parameter value (array, double word) and store in EEPROM $^{2)}$
12	Modify parameter value (array, word) and store in EEPROM $^{2)}$
13	Modify parameter value (double word) and store in EEPROM
14	Modify parameter value (word) and store in EEPROM

$^{1)}$ The desired element of the parameter description is specified in IND (2nd word)
$^{2)}$ The desired element of the indexed parameter is specified in IND (2nd word)

The meaning of the response identifier for response telegrams (inverter → master) is shown in the table below. The request identifier will determine which response identifiers are possible.

Table 2 - Response identifier (inverter → master) [1]

Response identifier	Description
0	No response
1	Transfer parameter value (word)
2	Transfer parameter value (double word)
3	Transfer descriptive element [1]
4	Transfer parameter value (array word) [2]
5	Transfer parameter value (array double word) [2]
6	Transfer number of array elements
7	Cannot process request, task cannot be executed (with error number)
8	No master control status / no parameter change rights for PARAMETER CHANNEL

[1] The desired element of the parameter description is specified in IND (2nd word)
[2] The desired element of the indexed parameter is specified in IND (2nd word)

Parameter index IND, 2nd word

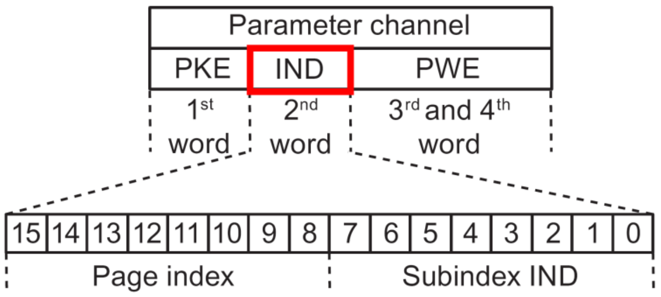


Fig. 6 – IND acyclic structure [1]

- The array subindex is an 8-bit value which is in acyclical data exchange mode (that we use for the communication with the inverter) transferred in the low-order byte (bits 0 to 7) of the parameter index (IND).
- The page index serves for setting the offset of the parameter number (PNU) that can only be in the range of 0 ÷ 1999. The coding of the offset is shown in the next picture and in the following table.

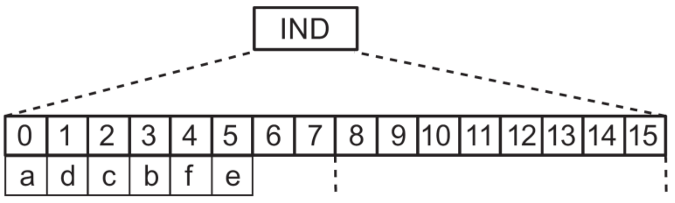


Fig. 7 – IND page index acyclic [1]

Table – Regulations for setting the page index [1]

Parameter range	Page index	Bit	Hex	+ PNU

	a	d	c	b	f	e	9	8	value	
0000 … 1999	0	0	0	0	0	0	0	0	0x00	0 – 7CF
2000 … 3999	1	0	0	0	0	0	0	0	0x80	0 – 7CF
4000 … 5999	0	0	0	1	0	0	0	0	0x10	0 – 7CF
6000 … 7999	1	0	0	1	0	0	0	0	0x90	0 – 7CF
8000 … 9999	0	0	1	0	0	0	0	0	0x20	0 – 7CF
…	…	…	…	…	…	…	…	…	…	…
32.000 … 33.999	0	0	0	0	0	1	0	0	0x04	0 – 7CF
…	…	…	…	…	…	…	…	…	…	…
64.000 … 65.999	0	0	0	0	1	0	0	0	0x08	0 – 7CF

Parameter value (PWE) 3rd and 4th word

For acyclic communication via USS, the number of PWEs can vary. For 16 bit values one PWE is required. If 32 bit values are exchanged, two PWEs are required. Since we set for our purposes the PKW to constant length to 4 words it means that the PWE area will always occupy 2 words. Then a 32-bit parameter value comprises PWE1 (high-order word, 3rd word) and PWE2 (low order word, 4th word). A 16-bit parameter value is transferred in PWE2 (low-order word, 4th word). PWE1 (high order word, 3rd word) must be set to 0 in this case.

Process data channel PZD

In this area of the telegram, process data (PZD) are continually exchanged between the master and slaves. It contains the signals required for the automation. Dependent on the direction the process data channel contains data for a request to the USS slaves or for a response to the USS master. In the requests are control words and setpoints for the slaves and in the responses are status words and actual values for the master. The number of PZD words in a USS-telegram is determined by parameter P2012. First two words are:

- Request to USS slave: Control word 1(STW1) and main setpoint (HSW)
- Response to USS master: Status word 1 (ZSW1) and actual value (HIW)

If P2012 is greater or equal to 4 the additional control word (STW2) is transferred as the 4th PZD word (default setting). The sources of all other PZDs are defined with parameter P2019 for a RS485 interface and P2016 for a RS232 interface.

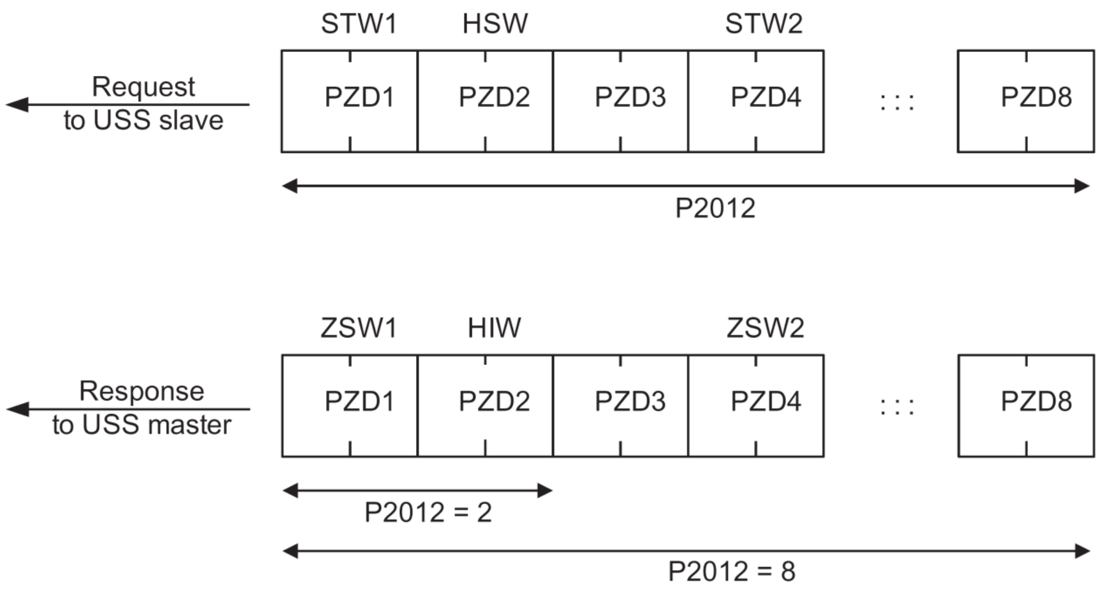


Fig. 8 – Process data channel structure [1]

Since we were also interested into continuous monitoring of torque and power of the electromotor we set the PZD length to 8 words and by the mean of BICO technology we linked some of the extra words so the channel structure was set as following:

PZD1 = STW1/ZSW1
PZD2 = HSW/HIW
PZD3 = not linked
PZD4 = STW2/ZSW2
PZD5 = not linked
PZD6 = not linked
PZD7 = r0031 = Act. filtered torque
PZD8 = r0032 = Act. filtered power

Actual values of parameters like setpoint, frequency, torque, power are read through normalized values. The range can be from -200% (8000_{hex}) to 199.99% (7FFF) of the reference value. It is therefore very important to set the reference values in expected range of the measurement to prevent exceeding. The range is coded as word (U16) it means that we get $400/2^{16}$ levels. That gives the resolution of 0.006% (fig. 9).

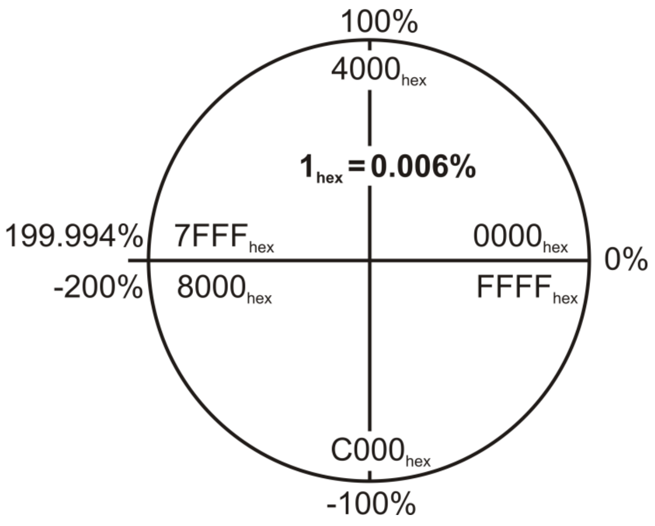
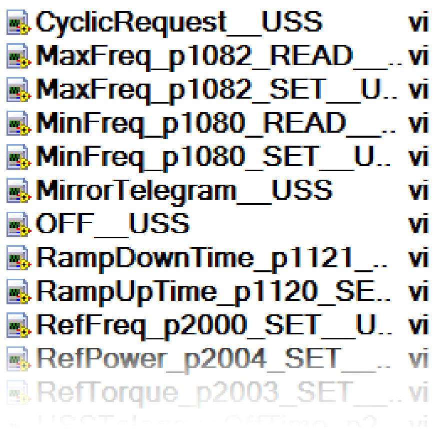


Fig. 9 – Valid range and U16 coding of normalized values

Implementing USS protocol to LabVIEW

As was already mentioned earlier there was a constant need for creation of native LabVIEW library covering the most common commands and routines and enabling easy creation of new ones. We decided to develop LabVIEW USS protocol library that would serve for purposes of our measurements on combustion engine. The final result was published for public use on National Instruments discussion forum – Motion Control and Motor Drives board.

It is important to emphasize that the programmed library expects the length of the PKW to be set to 4 words (P2013) and PZD to 8 words (P2012) otherwise the communication will fail. As described in previous chapter it also counts with PZD7 word to be linked to torque and PZD8 to power.



CyclicRequest__USS	vi
MaxFreq_p1082_READ__..	vi
MaxFreq_p1082_SET__U..	vi
MinFreq_p1080_READ__..	vi
MinFreq_p1080_SET__U..	vi
MirrorTelegram__USS	vi
OFF__USS	vi
RampDownTime_p1121_..	vi
RampUpTime_p1120_SE..	vi
RefFreq_p2000_SET__U..	vi
RefPower_p2004_SET__..	vi
RefTorque_p2003_SET__..	vi
[illegible]	vi

Fig. 10 –List of created commands and routines

We will describe the structure of the commands on one of the very basic ones – P1082, Maximal Frequency. Other commands have structure which more or less conforms to this one.

Each of the encapsulated commands has connectors (fig. 10) to enter the address of the inverter, VISA resource name that carries the information about the open communication channel, Index of the parameter we want to set and the value to be set. Error handling is presented.

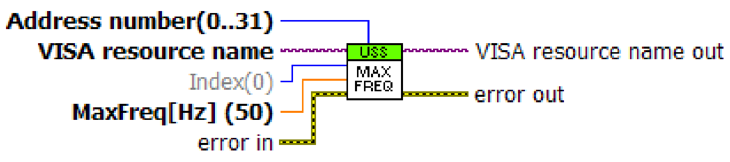


Fig. 11 – Connectors of single command

Inner command structure consists of six parts and can be seen on the fig. 11. The elements of the cluster marked (1) sets whether the PKW part should be present in the telegram. If set to false it automatically sends empty (=zeros) PKW channel. Next element is the request identifier according to the table 1. The number of the parameter, index and value of the parameter follow.

Cluster marked (2) sets whether the PZD part will be used. Next element determines whether the inverter should be switched ON/OFF. Following elements are the inversion of the setpoint and the desired value of the setpoint itself.

SubVI (3) handles creation of the telegram to be sent according to the protocol specifications stated in the previous chapters. LabVIEW Write primitive (4) writes the data to the port passed by the reference. The subVI marked with (5) continuously scans the port. It performs the check of the BCC, STX and the length of the answer. If no error occurs during reading and if the structure of the received telegram is correct it outputs the PKW and PZD otherwise it returns custom error describing the encountered problem. The time during which the subVI has to receive the valid response telegram (timeout value) can be set. The last subVI (6) parses the data and also checks for the inverter errors.

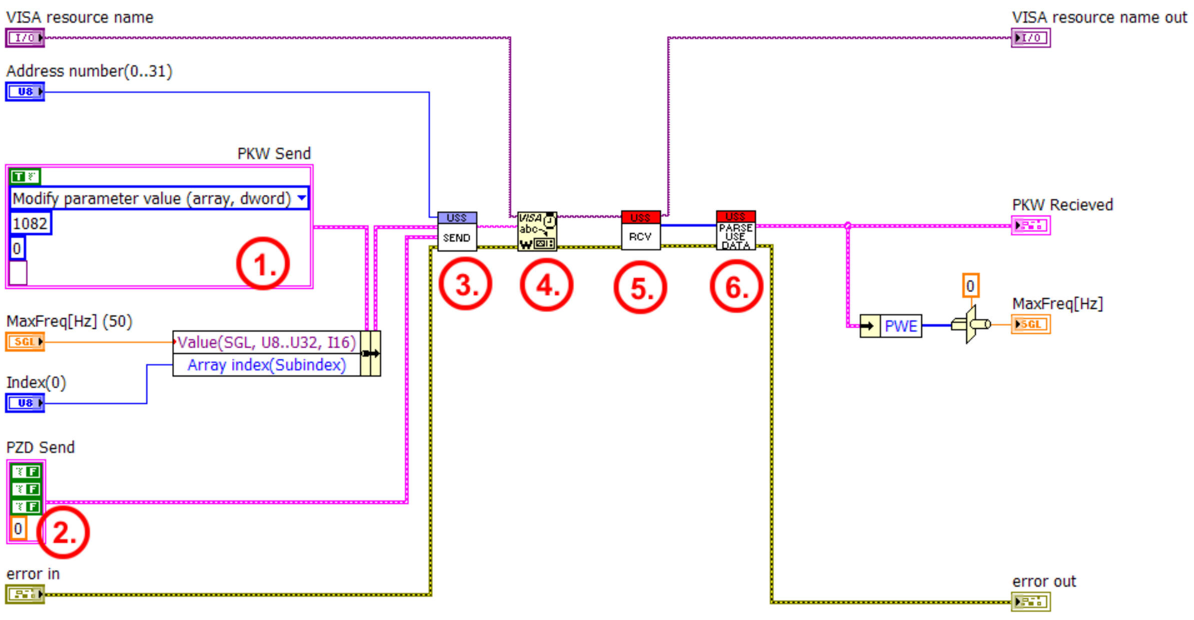


Fig. 12 – Description of the inner structure of USS commands

Slave response latency

During the period of testing programmed USS library we encountered also unexpected behavior of the response telegrams send by the inverter.

Communication with Siemens inverters is focused on cyclic communication. Since the typical scenario is for the master (typically a PLC) to continuously scan multiple slave devices, command data refresh rate is more important than response data latency. Rather than make the master wait for the slave to formulate a reply on every pass, the slave has the reply ready for the master based on the requested data in the previous pass. This way communication time is kept to a minimum.

This behavior unfortunately might complicate the acyclic communication when we just want to set some parameter of the inverter. The parameter will be changed correctly (if no fault or error occurs) but the immediate response of the slave will consist of the value of previously set/requested parameter. The same and even more confusing situation occurs when only requesting value of some parameter. It instead of sending the reply to requested parameter sends reply to the previous request. For example if we asked the value of parameter P1082 (maximum motor frequency) sending following telegram (PKW=4, PZD=8):

```
02 1A 00 64 3A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 46hex
```

64 $3A_{hex}$ = P1082

we would receive the answer with the value of parameter P1080 (minimum motor frequency) that we requested before:

```
02 1A 00 54 38 00 00 00 00 00 00 EB 31 00 00 00 00 06 E0 00 00 00 00 00 00 00 00 48hex
```

54 38_{hex} = P1080

This behavior is necessary to take into account when dealing with any implementation of the USS protocol. If you parse the entire reply you always know to which parameter apply the data but it won't probably be the same you set/asked.

Communication outages on RS485

Another problem that we came across was the random communication outages. The inverter time to time didn't reply when a telegram was sent to it most likely because it didn't receive the request or the request arrived distorted and thus it was refused.

We used RS485 (EIA-485) to exchange the telegrams. RS485 is a two-wire, half-duplex, multipoint serial communications channel. Since it uses a differential balanced line over twisted pair it can span relatively large distances (up to 1200m). Ideally, the two ends of the cable will have a termination resistor connected across the two wires. Without termination resistors, reflections of fast driver edges can cause multiple data edges that can cause data corruption. Termination resistors also reduce electrical noise sensitivity due to the lower impedance. Termination is usually required for long distances and high speeds. The length of the cable we used was about 4 meters and thus we didn't terminate the line at first. Nevertheless, random outages in the communication appeared. We thought that the termination might help and we switched ON the termination DIP switch on the CU240S and connected the termination 220Ω resistor across the two signal wires on the opposite end of the cable (schema depicted on figure 13).

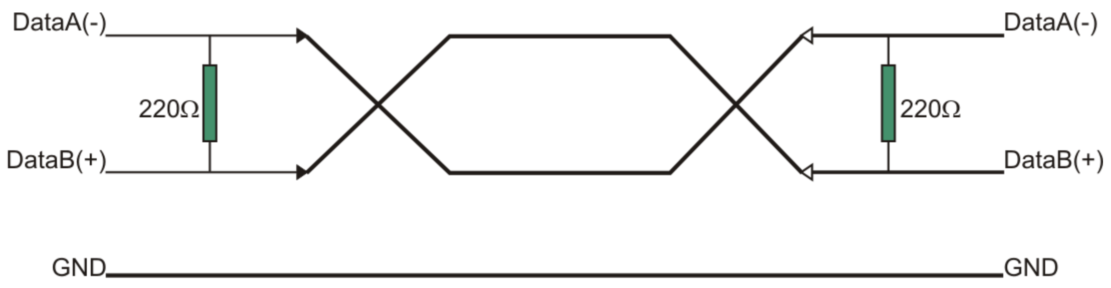


Fig. 13 – Termination schema

We also connected the bias resistors but none of those adjustments helped to prevent the communication outages. Since those problems appear randomly but rarely, we decided to implement in our application simple resending algorithm (not a part of the library!). The telegram is resent in predefined number of attempts (usually five). If none of those attempts succeeds the error is reported indicating that there is a major problem in the communication.

References

[1] *Siemens AG*, CU240S Operating Instructions, 10/2007, A5E00766042B AC

[2] *Siemens AG*, SINAMICS G120, Control Units CU240S, Parameter manual, Edition 05/2007

[3] *Siemens AG*, Specification, USS protocol, E20125-D0001-S302-A1-7600