MODBUSSLV - Driver of MODBUS/RTU Protocol for SLAVE Mode User's Manual

Doc. No. ENP4037
Version: 29-08-2005

1. MODBUSSLV - Driver of MODBUS/RTU Protocol for SLAVE Mode

1.1. Driver Use

The MODBUSSLV driver is used to access the process variables values of the **asix** system to other systems by means of a serial interface and the MODBUS protocol operating in RTU mode. In such connection the **asix** system operates as a subordinate device (SLAVE) of the MODBUS protocol. The MODBUSSLV protocol has the following types of variables implemented:

HR	(holding registers),
IR	(input registers),

and the following functions of the MODBUS protocol:

Read Holding Registers	(function 03),
Read Input Registers	(function 04),
Preset Single Register	(function 06),
Preset Multiple Registers	(function 16).

The function *Preset Multiple Registers* is limited to write the value of one process variable of the **asix** system.

1.2. Declaration of Transmission Channel

The full syntax of declaration of transmission channel operating according to the MODBUSSLV protocol is given below:

logical_channel_name=MODBUSSLV*, number, port, baud, bits, parity, stop*

where:

MODBUSSLV	- driver name;
number	- number of a remote device of the MODBUS network assigned to **asix**;
port	- name of the serial port;
baud	- transmission speed, max 115 kBd;
bits	- number of bits in a character;
parity	- parity check type;
stop	- number of stop bits.

EXAMPLE

The declaration of the logical channel named CHAN1, which works according to the MODBUSSLV protocol and with parameters as below:

- number - 4
- port - COM1
- transmission speed - 9600 Bd
- number of bits in a character - 8
- parity check type - parity check
- number of stop bits - 1

is as follows:

```
CHAN1 = MODBUSSLV, 1, COM1, 9600, 8, even, 1
```

The MODBUSSLV driver is loaded as a DLL automatically.

1.3. Addressing the Process Variables

The syntax of symbolic address which is used for variables belonging to the MODBUSSLV driver channel is as follows:

name, address [,scale] [,WITHOUT_STATUS |WITH_STATUS]

where:

name	- name of the **asix** process variable; it must have its equivalent among process variables declared in ASMEN files;
address	- type and number of the register by means of which the value of process variable is accessed; depending on the process variable type its value is transferred by means of one register (WORD or INT16 type variable) or two successive registers (FLOAT or DWORD type variable); there is a possibility to convert FLOAT type variables to a INT16 type variables;
scale	- it is used in case of converting the value of FLOAT type variables to a INT16 type number with simultaneous scaling (multiplication or division by power of 10); *scale* determines a sort of operation ('-' division) and exponent of power of 10; *scale* is used only when casting FLOAT type variables to INT16 was declared for the driver;
WITHOUT_STATUS	- variable value is transferred without status;
WITH_STATUS	- variable value is transferred with the status - the status occupies the next register after the register assigned to the variable.

EXAMPLE

A value of the variable X1 is transferred by means of the register HR1. The variable is of FLOAT type and its value is converted to INT16. Before the conversion the variable value is multiplied by 100. Independently of the value of the item WITH_STATUS the status (occupying the RH2 register) is transferred after the variable value.

```
X1,  HR1, 2, WITH_STATUS
```

The value of the variable X2 is transferred by the HR2 register. Depending on the variable type it occupies only the HR2 register (WORD or INT16 type variable) or HR2 and HR3 registers (FLOAT and DWORD type variable). The variable value is transferred without the status independently of the value of the item WITH_STATUS.

```
X2,  HR2, WITHOUT_STATUS
```

The value of the X3 variable is transferred by means of the HR3 register (if the variable is of WORD or INT16 type) or HR3 and HR4 registers (if the variable is of FLOAT or DWORD type). If the value of the item WITH_STATUS is YES, then the variable X3

status is transferred in the register HR4 (WORD or INT16 type variable) or HR5 (FLOAT or DWORD type variable).

X3, HR3

1.4. Driver Configuration

The MODBUSSLV protocol driver is configured by means of the section **[MODBUSSLV]** placed in the application INI file. Individual parameters are passed in separate items of the section. Each item has the following syntax:

item_name=[number [,number]] [YES|NO]

DATA_FILE = file_name

Meaning - the item allows to declare a file name in which the declarations of mapping the process variables of **asix** to the registers HR and IR of MODBUS. The section may include many DATA FILE items.
Default value - lack of default file.

Declaration of Transferring Statuses of Variables

WITH_STATUS = YES | NO

Meaning - the item allows to define globally the way of transferring the status of **asix** system process variables. The value of the WITH_STATUS item equal to YES signifies that status is sent with the value of the **asix** system process variable. The status is transferred in a register following the last register assigned to the process variable.
Default value - by default, the value of the item WITH_STATUS is equal to NO.

If the variable is of DWORD type and if the following declaration is given:

X1, HR3

then the variable status is transferred in the register HR5 (the X1 variable value is transferred in HR3 and HR4 registers).

If the variable X2 is of WORD type and if the following declaration is given:

X2, HR3

then the variable status is transferred in the register HR4 (the X2 variable value is transferred in the register HR3).

The WITH_STATUS item value equal to NO signifies that the process variable status is not transferred. It is possible to force the global settings by giving the component WITH_STATUS or WITHOUT_STATUS in the declaration of the process variable mapping. The declaration WITH_STATUS signifies that, irrespective of the WITH_STATUS value, the register following registers with the value of the considered

variable contains the variable status. The declaration WITHOUT_STATUS signifies that, irrespective of the WITH_STATUS value, the variable status is not transferred.

Casting Variables of FLOAT Type to INT16

CAST_TO_INTEGER = YES | NO

Meaning - the item equal to YES allows converting the values of FLOAT type to INT16 type. Before converting it is possible to scale the FLOAT value by multiplication by a power of 10. The power value is given optionally in the declaration of the ASMEN variable (component *scale*) mapping.

Default value - by default, the CAST_TO_INTEGER item value is equal to NO.

The declaration:

X1, HR1, -2

signifies that the value of X1 process variable will be divided by 100, and then it will be converted to a INT16 type number accessed in the register HR1.

The declaration:

X2, HR2, 3

signifies that the value of the X2 process variable will be multiplied by 1000, and then it will be converted to a INT16 type number accessed in the HR2 register.

Frequency of Refreshing Contents of Registers

DATA_UPDATE_PERIOD = number

Meaning - the values of registers are currently updated by reading data from ASMEN. The item determines a refreshing cycle (in seconds).

Default value - by default, the refreshing cycle is equal to 1 second.

Time of Answer Preparation

REPLY_TIMEOUT = number

Meaning - time of elaborating an answer (in milliseconds) is controlled by the value of the item REPLY_TIMEOUT. If in the time period passed by the REPLY_TIMEOUT item the driver is not able to prepare data required by the MASTER, then it does not send any reply.

Default value - by default, the answer preparation time is set to 200 milliseconds.

Maximal Number of Registers Given in a Query

NUMBER_OF_REGISTERS_PER_TELEGRAM = number

Meaning	- the item determines a maximal number of registers that may be asked by MASTER in one query. If the number of registers specified in this item is too large, then the driver sends an answer of the EXCEPTION type with the code 4 (SLAVE DEVICE FAILURE).
Default value	- by default, it is allowed to send queries including up to 128 registers.

LOG_FILE=file_name

Meaning	- the item allows to define a file to which all diagnostic messages of the MODBUSSLV driver are written. If the item does not define the full path, then the log file is created in the current directory. The log file should be used only while the **asix** start-up.
Default value	- by default, the log file is not created.

1. MODBUSSLV - DRIVER OF MODBUS/RTU PROTOCOL FOR SLAVE MODE 3

1.1. DRIVER USE 3
1.2. DECLARATION OF TRANSMISSION CHANNEL 3
1.3. ADDRESSING THE PROCESS VARIABLES 4
1.4. DRIVER CONFIGURATION 5