Soft-Starter SSW-07/SSW-08 Serial Communication Manual

02/2008

Series: SSW-07/SSW-08
Language: English
Document: 0899.5803 / 02

Summary

General Information

- Read this manual before installing or operating the Soft-Starter SSW-07/SSW-08.
- All information and safety notice included in this Manual must be strictly followed.
- To ensure physical integrity during operation and avoid damage to equipment and motors driven by the Soft-Starters SSW-07/SSW-08, provide electromechanical safety devices.

Safety Information

- Follow strictly all information given in this Manual relating to the cable interconnection of the two interfaces for the serial communication.
- Electronic boards are fitted with components that are sensitive to electrostatic discharge. When these boards are handled, take following cares:
  - Do not touch the components or connections (connectors) directly with the hands. When this is unavoidable, touch before a grounded metallic object or adopt proper grounding procedures.

Definitions

Used Terms

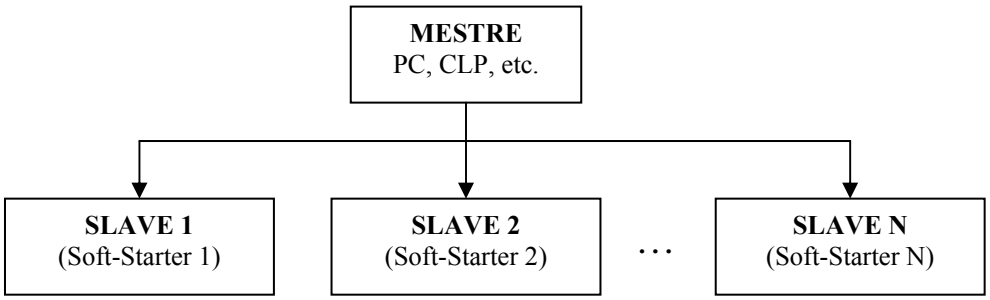
- Parameters: are those existing in the Soft-Starters SSW-07/SSW-08 and that can be displayed or changed through the Human-Machine-Interface (HMI).
- Basic Variables: internal Soft-Starter SSW-07/SSW-08 values that can be accessed only through the serial communication. The basic variables are used for monitoring the status, commands and equipment identification.
- Registers: are internal memory addresses of the Soft-Starter. These registers can be used for representing the basic variables and the parameters.
- EEPROM: is the non-volatile memory that saves the Soft-Starter SSW-07/SSW-08 parameter even when equipment is switched off.

Numerical Presentation

- Decimal numbers are represented by means of digits without suffix.
- Hexadecimal numbers are represented through the letter ’h’ after the number.

1. Introduction

The main purpose of the serial communication is the physical connection between two or more equipments on the network configured as follows:



By using this interface, the network master can request several services from every slave connected on the network. These services may be:

- IDENTIFICATION:
  - Equipment Type (frequency inverter, servoconverter, soft-starter)
  - Status monitoring
  - Error read

- PARAMETER SETTING
  - Parameter read (current, voltage, etc.)
  - Parameter write for equipment configuration

- COMMANDS
  - Enable / Disable
  - General Disable
  - *Error Reset*

The Soft-Starter SSW-07/SSW-08 uses the Modbus-RTU protocol for the communication through the serial interface. This protocol allows the Soft-Starter SSW-07/SSW-08 integration into different systems, enabling the connection of several equipments, such as:

- PC (master) for the parameter setting of one or various Soft-Starters SSW-07/SSW-08 simultaneously.
- SDCD monitoring Soft-Starter SSW-07/SSW-08 variables and parameters.

2. Soft-Starter SSW-07/SSW-08 Parameter Setting

Please find below the description of the parameter related to the serial communication and the operation via Modbus-RTU Protocol of the Soft-Starter SSW-07/SSW-08.

2.1. P308 – Soft-Starter SSW-07/SSW-08 Address on the Network

Each slave on the Network shall have a different address, so the master is able to sent the desired message to the specific slave on the network. This parameter allows setting the Soft-Starter SSW-07/SSW-08 address on the network.

Adjustable Range	Factory default	Access
1 ... 247	1	Read/Write

The use of a repeater is required if more than 30 equipments are used the same communication network.

2.2. P312 – Serial Protocol Type and communication rate

The Soft-Starter SSW-07/SSW-08 has one of the following options for the serial communication through product serial interface:

Adjustable Range	Factory default	Access
1 = Modbus-RTU, 9600 bit/s, no parity 2 = Modbus-RTU, 9600 bit/s, odd parity 3 = Modbus-RTU, 9600 bit/s, even parity 4 = Modbus-RTU, 19200 bit/s, no parity 5 = Modbus-RTU, 19200 bit/s, odd parity 6 = Modbus-RTU, 19200 bit/s, even parity 7 = Modbus-RTU, 38400 bit/s, no parity 8 = Modbus-RTU, 38400 bit/s, odd parity 9 = Modbus-RTU, 38400 bit/s, even parity	1	Read/Write

It is required that all equipments, operating on the same network, have the same communication configuration.

2.3. P313 – Action for Communication Error

This parameter allows programming an action that the drive shall adopt when a communication error is detected.

Adjustable Range	Factory default	Access
0 = No action 1 = Disable 2 = General disable 3 = Changes to LOC	1	Read/Write

- **0 – No action**: if some of the above mentioned error is detected, the SSW-07/SSW-08 remains in the current status and only the detected error is displayed.
- **1 – Disable**: SSW-07/SSW-08 will be disabled via voltage ramp when a communication error is detected.
- **2 – General Disable**: in this option, the starter disconnects the motor power supply and the motor will stop through inertia.
- **3 – Changes to LOC**: when the Soft-Starter SSW-07/SSW-08 is operating in remote mode and a communication error is detected, it changes automatically to local mode.

For the serial interface, only the t*imeout* error during the reception (E28 – Serial Communication is inactive) will be considered as communication error. The *timeout* during the reception is set through the Parameter P314.

The Error LED of the communication module indicates the E28 error. The error E28 is active when the LED is flashing.

NOTE!
The commands Disable and Change to LOC can be executed only when these commands are controlled via serial communication. This setting is executed through the Parameters P220, P229 e P230.

2.4. P314 – Time for *timeout* during the message reception

This function allows programming the time for the *timeout* detection during the message reception. The value 0 (zero) disables this function.

When the drive is controlled via serial communication and a communication problem to the Master is detected (cable rupture, voltage drop, etc.), it will not be possible to send a command to the serial for the equipment disable. In application where this will be a problem, you can set P314 to a longer interval within the SSW-07/SSW-08 can receive a valid serial message. Otherwise SSW-07/SSW-08 will interpret this as a serial communication error.

Adjustable Range	Factory default	Access
0 = Disabled function 0 ... 999 seconds	0	Read/Write

When this time has been programmed, however the SSW-07/SSW-08 remains for longer time than the programmed one without receiving valid serial messages, it will display E28 and it will adopt the action programmed in P313. After the communication is restored again, the E28 display will be deleted.

NOTE!

- When this function is disabled, you must ensure that the network master sends messages periodically to the slave, always considering the set time and thus preventing the timeout error detection during the communication.
- The detection of E28 will also reset the values of the basic variables 8 (see item 4.2).

2.5. P220 – Local/Remote Selection

This function enables programming the command source that controls the Local/Remote mode selection.

Adjustable Range	Factory default	Access
0 = Always Local 1 = Always Remote 2 = Keypad (standard is local) 3 = Keypad (standard is remote) 4 = DI1...DI3 5 = Serial (standard is local) 6 = Serial (standard is remote) 7 = Fieldbus (standard is local) 8 = Fieldbus (standard is remote)	3	Read/Write

If the control of the operating mode via serial is preferred, you must set this parameter to 5 or 6. The display "standard local" or "standard remote" informs which operation mode shall be activated after the equipment has been started.

2.6. P229 – Command Selection – Local Mode

This function allows programming the command source of the Soft-Starter SSW-07/SSW-08 when the Soft-Starter SSW-07/SSW-08 has been set to Locale mode.

Adjustable Range	Factory default	Access
0 = Keypad 1 = DI 2 = Serial 3 = Fieldbus	0	Read/Write

If the command control via serial communication in local mode is desired, you must set Parameter P229=2.

2.7. P230 – Command Selection – Remote Mode

This function enables selecting the Soft-Starter SSW-07/SSW-08 command source when the SSW-07/SSW-08 has been set to Remote Mode.

Adjustable Range	Factory default	Access
0 = Keypad 1 = DI 2 = Serial 3 = Fieldbus	1	Read/Write

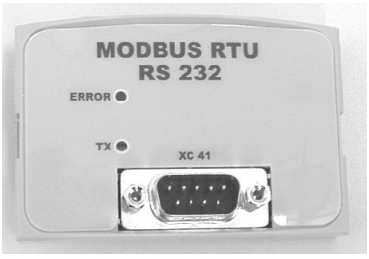
If the command control via serial communication in local mode is desired, you must set P230 = 2.

3. Interface Description

3.1. RS-232

To use the RS-232 with the Soft-Starter SSW-07/SSW-08 is necessary to install the Modbus RTU RS-232 Kit.

For more information see the RS-232 Installation Guide.

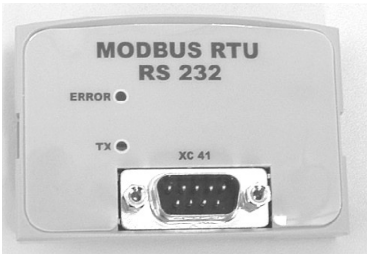


This interface enables the connection of the master to a Soft-Starter SSW-07/SSW-08 (point-to-point) at distances up to 10m. For the communication with the master you must use one wire for the transmission (TX), one wire for the reception (RX) and one wire for the reference (PE).

3.2. RS-485

To use the RS-485 with the Soft-Starter SSW-07/SSW-08 is necessary to install the Modbus RTU RS-485 Kit.

For more information see the RS-485 Installation Guide.



When the interface RS-485 is used, the master can control several drives connected to the same bus. The Modbus-RTU protocol allows the connection of up to 247 slaves (1 for each address), provided also signal repeaters are used along the bus.
This interface has a good noise immunity allowing cable lengths up to 1000 m.

3.2.1. Use of the RS-485 Kit for the SSW-07/SSW-08

The RS-485 Kit of the Soft-Starter SSW-07/SSW-08 comprises basically an interface board and the respective product installation instructions.

Following procedures should be adopted during the network installation when this interface is used:

- For the transmission the signals A and B use a shielded twisted pair of wire. These signals must be connected to the terminals A and B.
- The terminal COM is used for the connection of the reference signal to the RS-485 circuit. This connection is not required when this signal is not used.
- All RS-485 network connected devices must be grounded properly, preferably to the same ground point. Also the cable shield must be grounded. The shielding can be grounded at the connector XC42. If the shielding is grounded in another point, use the COM connection for the shielding.
- The network cable must be laid separately (if possible) distant from the power supply cables.
- Terminating resistor must be provided on the first and last device connected to the main bus. The RS-485 interface board is already fitted with witches for enabling this resistor. For this, set both switches to 'on'.

4. Accessible Data via Serial Communication

Some data can be accessed through the Soft-Starter SSW-07/SSW-08 serial interface for parameter setting, command and monitoring. On principle these data can be divided into two groups: parameters and basic variables.

4.1. Parameter of the Soft-Starter SSW-07/SSW-08

The parameters are those that are available though the HMI of the Soft-Starter SSW-07/SSW-08. Practically all Soft-Starter parameters can be accessed via serial communication and through these parameters you can program how this equipment will operate, as well as monitor all important information for the application, as current, errors, etc.

For complete parameter listing, please refer to Soft-Starter SSW-07/SSW-08 Programming Manual.

4.2. Available basic Variables for the Soft-Starter SSW-07/SSW-08

The basic variables are internal Soft-Starter SSW-07/SSW-08 values that can be accessed only through the serial product interface. Through these parameters you can monitor the Soft-Starter status as well as send enable, *reset* commands, etc.

Each basic variable represents a register (16 bits). The Soft-Starter SSW-07/SSW-08 enables following basic parameters:

4.2.1. Basic Variable 1

- *Variable*: VB01 – Soft-Starter SSW-07/SSW-08 Status.
- *Access*: read-only
- *Description*: indicates the Soft-Starter SSW-07/SSW-08 status. Each bit of this word provides a different indication:

Bit	Description
Bit 0	0 = motor stopped. 1 = motor running.
Bit 1	0 = when general disable due to any means. 1 = when general enable due to any means.
Bit 2	Reserved
Bit 3	0 = is not accelerating. 1 = during the whole acceleration process.
Bit 4	0 = is not in current limit. 1 = current limit.
Bit 5	0 = no full voltage is supplied to the motor. 1 = full voltage is supplied to the motor.
Bit 6	Reserved
Bit 7	0 = is not decelerating. 1 = during the whole deceleration process.
Bit 8	0 = local. 1 = remote.
Bit 9	0 = is not in DC braking. 1 = during DC barking process (Function not implemented in Software Version 1.3x)
Bit 10	0 = direction of rotation is not reverted. 1 = during the reversal process of the direction of rotation. (Function not implemented in Software Version 1.3x)
Bit 11	0 = CW. 1 = CWW. (Function not implemented in Software Version 1.3x)
Bit 12	0 = with open bypass. 1 = with closed bypass.
Bit 13	Reserved
Bit 14	0 = without power supply. 1 = with power supply.
Bit 15	0 = no error. 1 = with error.

4.2.2. Basic Variable 3

- *Variable*: VB03 – commands
- *Access*: read and write
- *Description*: allows commanding the Soft-Starter SSW-07/SSW-08 via serial communication.

This Word has 16 bits, but only the first 8 bits has a function. Each bit has the effective value of the command to be executed.

Bit	Description
Bit 0	0 = stopping by ramp. 1 = running by ramp.
Bit 1	0 = general disable 1 = general enable.
Bit 2	Reserved
Bit 3	0 = CW. 1 = CWW. (Function not implemented in Software Version 1.3x)
Bit 4	0 = local. 1 = remote.
Bit 5	Reserved
Bit 6	Reserved
Bit 7	0 = no command. 0 → 1 = executes *reset* (when in error status).

Always a command is sent to the Soft-Starter SSW-07/SSW-08, it will execute the command when it has been programmed to recept command via serial communication. This programming is performed through the following parameters:

- P220 - local / remote source selection.
- P229 – Command selection in local mode.
- P230 - Command selection in remote mode.

These commands must be set to the option "Serial" always the command should be executed via network. The *reset* command can be executed via network even when this parameter setting ha been performed, but only when the Soft-Starter SSW-07/SSW-08 is in error status.

NOTE!

- Communication board errors (E28, E29 or E30) cannot be reset in this way, since they depend on settings outside the values sent via network an in this condition the SSW-07/SSW-08 cannot establish communication with the network.
- So if you try to send a command via network that cannot be executed by the SSW-07/SSW-08 (for instance, a command that has not been programmed to operate via serial communication), this command will not be executed.

4.2.3. Basic Variable 8

- *Variable*: VB08 – commands for the digital outputs
- *Access*: read and write
- *Description*: it allows commanding the available Soft-Starter SSW-07/SSW-08 relay outputs. This Word has 16 bits, but only the first two bits has a function:

Bit	Description
Bit 0	0 = deactivates the RL1 relay output. 1 = activates the RL1 relay output.
Bit 1	0 = deactivates the RL2 relay output. 1 = activates the RL2 relay output
Bit 2 ... 15	Reserved

To command the digital outputs via serial communication, you must set its functions to the option "Serial", at the parameters P277 and P278. When the output is not being controlled via serial communication, the value received at the corresponding bit will be discarded.

When an error with the network master is detected (E28) the values of the digital outputs will be reset.

4.3. Changing of Parameters and Basic Variables

There are some peculiarities relating to the serial access of the Soft-Starter SSW-07/SSW-08 parameters and basic variable access

- There is no password for the access via serial communication. The parameter can be changed independent if the password is active or not.
- The value of P000 is not saved in the non-volatile equipment memory (via HMI it is saved).
- The Parameters P200 and P215 cannot be accessed via serial communication.
- When the run command of the basic variable is sent during the activation time of P630, the command will no be accepted, and the drive does not answer to the error.

5. Modbus-RTU Protocol

The Modbus Protocol has been developed in 1979. Today the Modbus Protocol is an open protocol widely used by many equipment manufacturers
The Modbus-RTU communication development of the Soft-Starter SSW-07/SSW-08 has been based on the following documents:

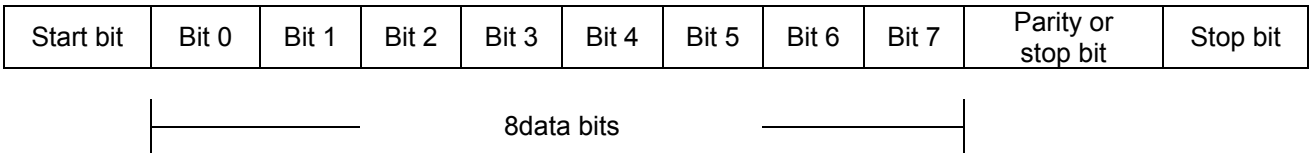
- MODBUS Protocol Reference Guide Rev. J, MODICON, June 1996.
- MODBUS Application Protocol Specification, MODBUS.ORG, May 8th 2002.
- MODBUS over Serial Line, MODBUS.ORG, December 2nd 2002.

These documents define the message format used by the elements that are part of the Modbus network, the services (or functions) that can be enabled via network, and how these elements are exchanged on the network.

5.1. Transfer Modes

In the protocol specification are defined two transfer modes: ASCII e RTU. The modes define the way in which these message bytes are transferred. It is not allowed to use two transfer modes on the same network.

The Soft-Starter SSW-07/SSW-08 uses only the RTU mode for the message transfer. The bytes are transferred in hexadecimal format, where each transferred byte has a 1start bit, 8 data bits, 1 parity bit (optional) and 1 stop bit (2 stop bits, if no parity bit is used). The byte format configuration is made through the parameter P312.



5.2. Message structure in RTU mode

The Modbus-RTU network uses the master-slave system for the message exchange. The Modbus-RTU network can have up to 247 slaves, but only one master. Every communication starts with the master making a request to a slave and the slave answers according to the request. In both messages (answer and response), the used structure is the same: Address, Function Code, Data and CRC. Only one data filed can have variable length, depending on what is being requested.

Master (requesting telegram):

Address (1 byte)	Function (1 byte)	Requesting Data (n bytes)	CRC (2 bytes)

Slave (answer message):

Address (1 byte)	Function (1 byte)	Requesting Data (n bytes)	CRC (2 bytes)

5.2.1. Address

The master starts the communication by sending a byte with the slave address to which the message should be sent. The slave also starts the answer (message) with its own address. The Master can also send a message to the zero (0) address, which means that the message is sent to all network slaves (*broadcast*). In this case, no slave will answer to the Master.

5.2.2. Function Code

This field has an only Byte where the Master specifies the service of function type requested from the slave (read, write, etc.) According to the protocol, each function is used for accessing a specific data type.

For the Soft-Starter SSW-07/SSW-08, the data relating to the parameters and basic variables are available as registers of *holding* type (referenced from the address 40000 or '4*x*' on).

5.2.3. Data Field

Data field with variable length. The format and the content of this field depend on the used function and on the transmitted values. This field is described jointly the functions (see item 5.4).

5.2.4. CRC

The last part of the message is the filed for checking the transmission errors. The used method is the CRC-16 (Cycling Redundancy Check). This field is formed by two Bytes, where the least significant Byte (CRC-) is transmitted first, after the most significant Byte (CRC+) is transmitted. The CRC calculation form is described in the protocol specification, however the information for its implementation is given in the Appendix A and C.

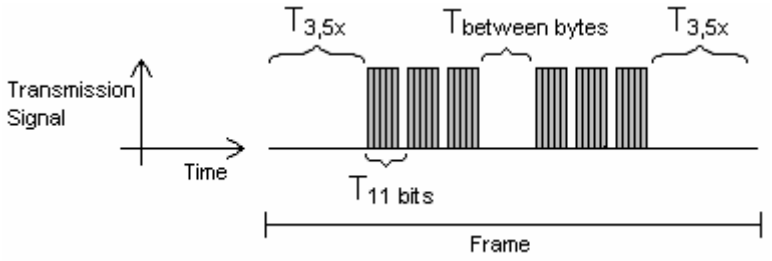
5.2.5. Times between Messages

There is no specific character in the RTU mode that indicates the beginning or end of a message. Thus, the only indication for the beginning or the end of a new message is the data transmission absence in the network by 3.5 times the time required for transmission of one data byte (11 bits).

Thus if a message if a message is initiated after elapsing of the minimum time required without transmission, the network elements assume that the received character represents the beginning of a new message. In similar mode, after this time has elapsed the network elements will assume that the message has been ended.

If during transmission of a message, the time between the bytes is longer than the required time, the message will be considered invalid, since the drive will discard the already received bytes and will mount a new message with the bytes that are being transmitted.

The table below shows the time for the different communication rats.



Communication rate	T 11 bits	T 3,5x
9600 bits/sec	1,146 ms	4.010 ms
19200 bits/sec	573 µs	2.005 ms
38400 bits/sec	573 µs	2.005 ms

- T 11 bits = Time to transmit one word of the message.
- T between bytes = Time between bytes (can not be longer than T 3,5x).
- T 3,5x = Minimum interval to indicate the begin and the end of the Message (3,5 x T 11bits).

For communication rates higher than 19200 bits/s will be considered the same time than those used for 19200 bits/s.

5.3. Operation of the Soft-Starter SSW-07/SSW-08 on the Modbus-RTU Network

The Soft-Starter SSW-07/SSW-08 will have following characteristics when operated on the Modbus-RTU network:

- Network connection via serial interface RS-232 or RS-485 (see item 3).
- Addressing, communication rate and bytes format defined through the parameters (see item 2).
- It allows the parameter setting and the equipment control through the access to parameters and basic variables.

5.3.1. Available Functions and Response Times

In the Modbus-RTU protocol is defined the functions used for accessing different types of registers. In the Soft-Starter SSW-07/SSW-08, both parameters and basic variables are defines as being *holding* type registers. In addition to those registers, you can also access the internal bits and the monitoring bits, designated as *coils* directly. Following services (or functions) are available for accessing these bits and register):

- *Read Coils*
  Description: reading of internal register blocks or coils.
  Function code: 01.
  Response time: 5 to 20 ms.

- *Read Holding Registers*
  Description: reading of register blocks of *holding* type
  Function code: 03.
  Response time: 5 to 20 ms.

- *Write Single Coil*
  Description: reading of a single internal bit or coil.
  Function code: 05.
  Response time: 5 to 20 ms.

- *Write Single Register*
  Description: reading of a single register of *holding* type
  Function code: 06.
  Response time: 5 to 20 ms.

- *Write Multiple Coils*
  Description: writing of internal bit blocks or coils.
  Function code: 15
  Response time: 5 to 20 ms.

- *Write Multiple Registers*
  Description: writing in register blocks of *holding* type
  Function code: 16.
  Response time: 20 ms for each written register.

- *Read Device Identification*
  Description: Identification of the drive model.
  Function code: 43.
  Response time: 5 to 20 ms.

5.3.2. Data Addressing and Offset

The Soft-Starter SSW-07/SSW-08 data addressing is realized with an offset equal to zero, which means that the address number is equal to the data number. The parameters are available from address 0 (zero) on, whilst the basic variables are available from the address 5000 on. In the same way, the status bits are made available from

address 0 (zero) and the control bits are made available from address 100 on. Table below shows the addressing of the parameters and basic variables:

PARAMETERS		
Parameter Number	Modbus Address	
	Decimal	Hexadecimal
P000	0	0x0000
P001	1	0x0001
⋮	⋮	⋮
P101	101	0x0065
⋮	⋮	⋮

BASIC VARIABLES		
Number of the Basic Variable	Modbus Address	
	Decimal	Hexadecimal
V01	5001	0x1389
⋮	⋮	⋮
V08	5008	0x1390

STATUS BITS		
Bit Number	Modbus Address	
	Decimal	Hexadecimal
Bit 0	00	00h
Bit 1	01	01h
⋮	⋮	⋮
Bit 15	15	0Fh

COMMAND BITS		
Bit Number	Modbus Address	
	Decimal	Hexadecimal
Bit 100	100	64h
Bit 101	101	65h
⋮	⋮	⋮
Bit 107	107	6Bh

The Status Bits (0 to15) have the same function of each bit of the basic variable 1 (see item 4.2.1), whilst the command bits (100 to 107) have the same function of the least significant bits of the basic variable 3, without required the us of the mask for the SSW-07/SSW-08 control (see item 4.2.2).

NOTE!
All registers (parameters and basic variables) are considered as *holding* type registers. Depending on the used master, these registers are referenced from the basic address 4000 or 4*x* on. In this case, the address of a parameter or basic variable that should be set

on the master is the address shown in the table above added to the basic address. In the same way, the bits are referenced from 0000 or 0*x* o, designated as *coils*. For more information on how access the registers of *holding* type and *coils*, refer to the documents about the master.

5.4. Detailed Function Description

In the section is given a detailed of the functions that are available in the SSW-07/SSW-08 for the Modbus-RTU communication. For the message preparation, please consider following:

- The values are always transmitted as hexadecimal values.
- The address of one data, the data number and the value of the register are always represented through 16 bits. Thus these fields are transmitted by using two bytes (*high* and *low*).
- The messages, both for enquiry and for response, cannot be longer than 256 bytes.
- The transmitted values are always integer numbers, independent if they are represented by decimal place or not. Thus the value of 9.5 will be transmitted as 95 via serial communication. Relating to the used resolution for each parameter, refer to the SSW-07/SSW-08 manual.

5.4.1. Function 01 – Read Coils

It reads the content of an internal group of bits that must be compulsory in a numerical sequence. This function has the following structure for the read and response messages (the values are always hexadecimal, and each field represents one byte):

Query (Master)	Response (Slave)
Salve address	Slave address
Function	Function
Initial bit address (byte high)	Byte Count Field (number of data bytes)
Initial bit address (byte low)	Byte 1
Number of bits (byte high)	Byte 2
Number of bits (byte low)	Byte 3
CRC-	etc...
CRC+	CRC-
	CRC+

Each response bit is placed at a position of the bytes data sent by the slave. The first byte receives the first 8 bits from the initial address indicated by the master. The other bytes (if the number of the read bits is higher than 8) remain in the same sequence.
If the number of the read bits is not a multiple of 8, the remaining bits of the last byte should be filled out with 0 (zero).

Example: reading of the status bits for enable (bit 0) and general enable (bit 1) of the SSW-07/SSW-08 at the address 1 (supposing Enable inactive and General Enable active):

- Address: 1 = 01h (1 byte)
- Number of the initial bit: 0 = 0000h (2 bytes)
- Number of the read bits: 2 = 0002h (2 bytes)

Query (Master)		Response (Slave)	
Field	*Value*	*Filed*	*Value*
Slave address	01h	Slave address	01h
Function	01h	Function	01h
Initial Bit (high)	00h	Byte Count	01h
Initial Bit (low)	00h	Status of the bits 1 and 2	02h
Number of bits (high)	00h	CRC-	D0h
Number of bits (low)	02h	CRC+	49h
CRC-	BDh		
CRC+	CBh		

As the number of read bits in the example is smaller than 8, the slave required only 1 byte for the response. The value of the byte was 02h that, as binary value, will have the form 0000 0010. As the number of read bits is equal to 2, only the two less significant bits, that have the value 0 = enable and 1 = general enable are of interest. The other bits, as they did not be requested, are filled out with 0 (zero).

5.4.2. Function 03 – Read Holding Register

It reads the content of a group of registers that must be compulsorily in a numerical sequence. This function has following structure for the read and response messages (the values are always hexadecimal values, and each field represents one byte):

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Initial bit address (byte high)	Byte Count Field
Initial bit address (byte low)	Data 1 (high)
Number of bits (byte high)	Data 1 (low)
Number of bits (byte low)	Data 2 (high)
CRC-	Data 2 (low)
CRC+	etc.
	CRC-
	CRC+

Example 2: Read of the motor current as percentage (P002) and read of the motor current in Amperes (P003) of the Soft-Starter SSW-07/SSW-08 at the address 1 (supposing P002 = 50.0% and P003 = 40.0 A).

- Address: 1 = 01h (1 byte)
- Number of the first parameter: 2 = 0002h (2 bytes)
- Number of the read parameters: 2 = 0002h (2 bytes)
- Read value of the first parameter: 500 = 01F4h (2 bytes)
- Read value of the second parameter: 400 = 0190h (2 bytes)

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	01h	Slave address	01h
Function	03h	Function	03h
Initial register (high)	00h	Byte Count	04h
Initial register (low)	02h	P002 (high)	01h
Number of registers (high)	00h	P002 (low)	F4h
Number of registers (low)	02h	P003 (high)	01h
CRC-	65h	P003 (low)	90h
CRC+	CBh	CRC-	BBh
		CRC+	C1h

5.4.3. Function 05 – *Write Single Coil*

This function is used to write a value to a single bit (*Coil*). The bit value is represented by using two bytes, where FF00h represents the bit that is equal to 1, and 0000h represents the bit that is equal to 0 (zero). It has the following structure (the values are always hexadecimal, and each field represents one byte):

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Bit address (byte high)	Bit address (byte high)
Bit address (byte low)	Bit address (byte low)
Bit value (byte high)	Bit value (byte high)
Bit value (byte low)	Bit value (byte low)
CRC-	CRC-
CRC+	CRC+

Example 3: write of the *reset* command (bit 107) in a Soft-Starter at address 1.

- Address: 1 = 01h (1 byte)
- Bit number: 107 = 006Bh (2 bytes)
- Bit value: *reset* = 1, so the value to be written is FF00h

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	01h	Slave address	01h
Function	05h	Function	05h
Bit number (high)	00h	Bit number (high)	00h
Bit number (low)	6Bh	Bit number (low)	6Bh
Bit value (high)	FFh	Bit value (high)	FFh
Bit value (low)	00h	Bit value (low)	00h
CRC-	FDh	CRC-	FDh
CRC+	E6h	CRC+	E6h

Please note, that for this function, the response of the slave will be an identical copy of the query made by the master.

5.4.4. Function 06 – *Write Single Register*

This function is used to write a value to a single register. This function has following structure (values are always hexadecimal values, and each field represents one byte):

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Register address (byte high)	Register address (byte high)
Register address (byte low)	Register address (byte low)
Register value (byte high)	Register value (byte high)
Register value (byte low)	Register value (byte low)
CRC-	CRC-
CRC+	CRC+

Example 4: Write of a logic command (basic variable 3) with the ramp enable command and the general enable command for the Soft-Starter SSW-07/SSW-08 at address 3.

- Address: 3 = 03h (1 byte)
- Number of the variable: VB03, addressed in the register 5003 = 138Bh (2 bytes)
- Value of the variable: ramp enable → command in 1 (bit 0)
  General enable → command in 1 (bit 1)
  Thus the value for the command is = 0003h (2 bytes)

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	03h	Slave address	03h
Function	06h	Function	06h
Register (high)	13h	Register (high)	13h
Register (low)	8Bh	Register (low)	8Bh
Value (high)	00h	Value (high)	00h
Value (low)	03h	Value (low)	03h
CRC-	BCh	CRC-	BCh
CRC+	87h	CRC+	87h

Please note, that for this function, the response of the slave will be an identical copy of the query made by the master.

5.4.5. Function 15 – *Write Multiple Coils*

This function allows writing values for a bit group (*coils*) that must be in numerical sequence. This function can be also used to write a single bit (the values are always hexadecimal, and each field represents one byte).

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Initial bit address (byte high)	Initial bit address (byte high)
Initial bit address (byte low)	Initial bit address (byte low)
Number of bits (byte high)	Number of bits (byte high)
Number of bits (byte low)	Number of bits (byte low)
Byte Count Filed (number of data bytes)	CRC-
Byte 1	CRC+
Byte 2	
Byte 3	
etc...	
CRC-	
CRC+	
Slave address	

The value of each bit that is being sent is placed at a position of the data bytes sent by the master. The first byte receives the 8 first bits by starting from the initial address indicated by the master. The other bytes (if the number of inscribed bits is higher than 8) remain in sequence. If the number of inscribed bits is not a multiple of 8, the remaining bits of the last byte should be filled in with 0 (zero).

Example 5: write of the bits 100 and 101 for ramp enable and general enable of a Soft-Starter SSW-07/SSW-08 at the address 20

- Address: 20 = 14h (1 byte)
- Number of the first bit: 100 = 0064h (2 bytes)
- Number of bits: 2 = 0002h (2 bytes)
- Value of the bits: the two bits must be placed at 1, then the value will be = 03h (1 byte)

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	14h	Slave address	14h
Function	0Fh	Function	0Fh
Initial bit (byte high)	00h	Initial bit (byte high)	00h
Initial bit (byte low)	64h	Initial bit (byte low)	64h
Number of bits (byte high)	00h	Number of bits (byte high)	00h
Number of bits (byte low)	02h	Number of bits (byte low)	02h
Byte Count	01h	CRC-	97h
Value of the bits	03h	CRC+	10h
CRC-	2Eh		
CRC+	6Dh		

5.4.6. Function 16 – *Write Multiple Registers*

This function allows writing values to a register group that must be in numerical sequence. This function can also be used to write a single register (the values are always hexadecimal values and each field represents one byte).

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
Address of the initial register (byte high)	Address of the initial register (byte high)
Address of the initial register (byte low)	Address of the initial register (byte low)
Number of registers (byte high)	Number of registers (byte high)
Number of registers (byte low)	Number of registers (byte low)
Byte Count Filed (number of data bytes	CRC-
Data 1 (high)	CRC+
Data 1 (low)	
Data 2 (high)	
Data 2 (low)	
etc...	
CRC-	
CRC+	

Example 6: write of the value 2 on P313 and value 5 on P314, for a Soft-Starter SSW-07/SSW-08 at the address 15.

- Address: 15 = 0Fh (1 byte)
- Number of the first parameter: P313, addressed in the register 313 = 139h (2 bytes)
- Value of the first parameter: 2 = 0002 (2 bytes)
- Value of the second parameter: 5 = 0005h (2 bytes)

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	0Fh	Slave address	0Fh
Function	10h	Function	10h
Initial register (high)	01h	Register (high)	01h
Initial register (low)	39h	Register (low)	39h
Number of register (high)	00h	Value (high)	00h
Number of register (low)	02h	Value (low)	02h
Byte Count	04h	CRC-	91h
P313 (high)	00h	CRC+	17h
P313 (low)	02h		
P314 (high)	00h		
P314 (low)	05h		
CRC-	68h		
CRC+	6Ah		

5.4.7. Function 43 – *Read Device Identification*

Auxiliary function that allows to read the manufacturer name, the model and version of the product firmware. It has following structure:

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function
MEI Type	MEI Type
Read code	Conformity Level
Object number	More Follows
CRC-	Next object
CRC+	Number of objects
	Code of the first object
	Length of the first object
	Value of the first object (n bytes)
	Code of the second object
	Length of the second object
	Value of the second object (n bytes)
	etc...
	CRC-
	CRC+

This function permits reading three information categories: Basic, Regular and Extended and each category are formed by a group of objects. Each object is formed by a sequence of ASCII characters. For the Soft-Starter are available only basic information formed by three objects:

- Object 0x00 - VendorName: always 'WEG'.
- Object 0x01 - ProductCode: formed by the product code (SSW-07), plus the rated drive current (ex. 'SSW-07 85A').
- Object 0x02 - MajorMinorRevision: it indicates the drive firmware version, in 'VX.XX' format.

The read code indicates which information categories are being read and if the objects are accessed individually of by sequence.
In the example, the SSW-07 supports the codes 01 (basic information in sequence), and 04 (individual access to the objects).
The other fields for the SSW-07 have fixed values.

Example 7: read of the basic information in sequence, starting from the object 00h of a Soft-Starter SSW-07 at the address 1:

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	01h	*Field*	01h
Function	2Bh	Slave address	2Bh
MEI Type	0Eh	Function	0Eh
Read Code	01h	MEI Type	01h
Object Number	00h	Conformity Level	51h
CRC-	70h	More Follows	00h
CRC+	77h	Next Object	00h
		Number of Objects	03h
		Object Code	00h
		Object length	03h
		Object value	'WEG'
		Object Code	01h
		Object length	0Ch
		Object value	'SSW-07 85.0A'
		Object Code	02h
		Object length	05h
		Object value	'V1.20'
		CRC-	CBh
		CRC+	5Eh

In the example the Object Value has not been represented as hexadecimal value, but with corresponding ASCII characters. For instance, for the object 00, the ´WEG‘ value has been transmitted as being three ASCII characters, that as hexadecimal representation have the values 57h ('W'), 45h ('E') and 47h ('G').

5.4.8. Communication Errors

Errors can occur during the message transmission on network, or in the content of the received messages. Depending on the error type, Soft-Starter SSW-07/SSW-08 may answer or not to the master. When the master sends a message to a slave configured at determined network address, the slave wills not response to the master if:

- Error in the parity bit.
- Error in CRC.
- *Timeout* between transmitted bytes (3.5 times the time required for transmitting one byte).

In this case the master must detect a *timeout* error during the time he was waiting for the slave response. When message is received with success, the drive can detect problems during the message processing and send an error message by indicating the detected problem:

- Function is not valid (error code = 1): the requested function is not implemented for the equipment.
- Data address is not valid (error code = 2): the data address (parameter) does not exist.
- The data value is not valid (error code = 3): this error is detected in the following conditions:
  - Value is out of range.
  - Data write cannot be changed (read-only register).
  - Command is not enabled for serial execution.

NOTE!
It is important to identify in the master the type of error that has been detected so you are able to diagnose the problems during the communication process.

If some error has been detected, the slave must return a message to the master indicating the type of error that has been detected. The messages sent by the slave have the flowing structure:

Query (Master)	Response (Slave)
Slave address	Slave address
Function	Function (with the most significant bit at 1)
Data	Error code
CRC-	CRC-
CRC+	CRC+

Example 8: master requests to the slave at address 1 the write in parameter 89 (non existing parameter):

Query (Master)		Response (Slave)	
Field	*Value*	*Field*	*Value*
Slave address	0x01	Slave address	0x01
Function	0x06	Function	0x86
Register (high)	0x00	Error code	0x02
Register (low)	0x59	CRC-	0xC3
Value (high)	0x00	CRC+	0xA1
Value (low)	0x00		
CRC-	0x59		
CRC+	0xD9		

APPENDIXES

Appendix A - CRC calculation by using the tables

Please find below a function that uses de “C” programming language and that implements the CRC calculation for the Modbus-RTU protocol. The calculation uses two tables for supplying pre-calculated shift values required for the calculation. More information about the algorithm can be obtained in the referenced documents in 4.

```
/* Table of CRC values for high-order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40};

/* Table of CRC values for low-order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04,
0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8,
0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11, 0xD1, 0xD0, 0x10,
0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C,
0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0,
0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C,
0xB4, 0x74, 0x75, 0xB5, 0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54,
0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98,
0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40};

/* The function returns the CRC as a unsigned short type */
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg;                  /* message to calculate CRC upon    */
unsigned short usDataLen;                /* quantity of bytes in message     */
{
    unsigned char uchCRCHi = 0xFF;       /* high byte of CRC initialized     */
    unsigned char uchCRCLo = 0xFF;       /* low byte of CRC initialized      */
    unsigned uIndex;                     /* will index into CRC lookup table */
    while (usDataLen--)                  /* pass through message buffer      */
    {
        uIndex = uchCRCLo ^ *puchMsgg++; /* calculate the CRC                */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex};
        uchCRCHi = auchCRCLo[uIndex];
    }

    return (uchCRCHi << 8 | uchCRCLo);
}
```

Appendix B - CRC calculation by using the register shift

In this section is described the algorithm for the CRC calculation used in the Modbus-RTU communication through the register shift; More information about the algorithm can be found in the referenced documents in item 5.

The CRC calculation is started by loading a 16 bits variable (afterwards designated as CRC variable) with the value of 0xFFFF. After proceed according to the following routines:

1. Submit the first byte of the message (only the data bits - start bit, parity bit and stop bit are not used) to a logic XOR (OR exclusive) with the 8 least significant bits of the CRC variable, returning the result in the own CRC variable.
2. Then the CRC variables is shift one place to right, to the direction of the least significant bit, and the place of the most significant bit is filled in with 0 (zero).
3. After this shift, the *flag* bit (the bit that has been shifted outside the CRC variable) is analyzed, and:
   - ☑ If the bit value is 0 (zero), no measure is adopted.
   - ☑ If the bit value is 1, the variable content is submitted to a logic XOR with a constant 0xA001 value and the result is returned to the CRC variable.
4. Steps 2 and 3 are repeated till eight shifts are executed.
5. Steps 1 to 4 are repeated by using the next message byte till the complete message has been processed.

The end content of the CRC variable is the value of the CRC field that is transmitted in the end message. The least significant part is transmitted first (CRC-) and in sequence the most significant part is transmitted (CRC+).