# EARTH FAULT & SHORT CIRCUIT INDICATOR

## MODBUS USER MANUAL

REV01 - May 2017

# Contents

## *TABLES*



## *FIGURES*

# COMMUNICATION

Main unit is designed to communicate in serial with the other devices over the RS485 data link with Modbus[1] RTU protocol.

The device stores the data in the registers that it can use for communication and makes it possible to access them with the parameters of the protocol.

The device can communicate with different parities (even, odd, none) and baud rates (9600, 19200, 38400 kbps) options. The data length is 8 bits long.

In communication, there are some restrictions due to different reasons. It allows interpretation of errors in case of violation of these restrictions by returning error messages. It is advised that software that will communicate with AAGD should be able to take into account the restrictions and interpret the returns.

This document is about communication method with Modbus RTU protocol, register addresses of Earth Fault & Short Circuit Indicator and related restrictions.

## Modbus

Modbus is a serial communication protocol developed by Modicon for use with PLCs in 1979. Modbus can provide communication between 247 devices. The operational logic is based on data exchange between a master device and more devices(slaves) connected to the master device on the same network.

Modbus protocol has many versions; AAGD uses Modbus RTU among these versions.

### *Data Model*

The Modbus data model is created by separating data according to the distinguishing characteristic of the data. Accordingly, 4 basic data tables have occurred.

| Table Name | Data Type | Feature | Explanation |
|---|---|---|---|
| **Discrete Input** | Bit | Read-Only | It covers only the readable bits of the system. |
| **Coils** | Bit | Read and Write | It covers the exchangeable bits of the system. |
| **Analog Registers** | 16-bit | Read-Only | It covers the readable analog data of the system. |
| **Writable Registers** | 16-bit | Read and Write | Reserved modifiable registers of the system. |

Table 1: Modbus Data Model

**Discrete input values** hold values used as single-bit status indicators. Since there is no such status indicator variable in AAGD, there is not any defined discrete input in this version.

**Coils** are defined as user-changeable values. Relay values in AAGD are mapped as coils. These values are readable only because they cannot be changed by the user.

**Analog registers** are 16-bit long units and represent the units that are assigned to hold values which cannot be modified by the user. For AAGD, 164 analog registers in total are assigned. Some of the analog registers are divided into 2-bit registers by data coding method because the lengths are 32-bit (4 byte).

[1] Detailed information for modbus: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

**Writable registers** are 16-bit registers and hold values that can be changed by user. There are total of 12 writable registers for different variables in AAGD. Because the information stored here directly affects the operation of the device, writing requests should not be sent without being sure during the operation.

### *Data Codes*

Modbus generate data frames by encoding with "Big-Endian" representations for addresses and the datas. For the datas that do not fit in a byte, the big-bytes are send as the first bytes.

**Example:**

| Register Size | Value | MSB | LSB |
|---|---|---|---|
| 16-bit (2 byte) | **0x2450** | first byte **0x24** | last byte **0x50** |

Figure 1: "Big-Endian" Representation

### *Addressing Model*

Separate registers are used for each data model in AAGD. Therefore, there is an addressing starting from "0" to "65535" for each data model. This addressing model is configured in accordance with the "IEC-61131 object" standard.

Necessary addresses and data types in AAGD are shown on the relevant tables.

### *Functions*

The device interprets the data block sent by the mainframe and generates an appropriate response. The second byte of sent data block gives information that will be performed by the device.

The vast majority of the basic functions that provide Modbus communication are pre-defined and standardized. Some of these functions can be used for ESI, but others are not available because they indicate non-existent registers and functional features.

02 function is used to read discrete inputs but it has an invalid function value because there are no discrete inputs to be read on the device. Modbus can provide up to 65 predefined functions, but not all of these functions are standard, as well as allowing user-defined functions after 65. Function values can have up to 255 values, starting with 01, allowing a byte in a data frame.

Functions that can be used in the device:

**Function 03 –** The function used to read the **Writable Registers** can be used to read a single register or multiple register. The return values return as a data frame.

**Function 04 –**The function used to read the **Analog Registers** allows single or multiple reads.

**Function 06 – Single Register Writing** function is used to assign a new value to one of the registers that can be written.

**Function 16 – Multiple Register Writing** function is used to give new values to more than one of registers that can be written. The register address to be written has to be sequential.

The functions except above functions are not included in this user manual because they are not used for this device.

### *Data Frame*

Modbus provides serial communication by sending and receiving data as block. The first byte of this block carries the address information. So, devices that communicate with Modbus must have an address value less than 255(0xFF), which is the maximum value that a byte can represent.

The second byte of data frame carries function information. The datas received after the function byte are evaluated in different ways according to the function information.

The last two cells (2 bytes) of each data frame are reserved for CRC (Cyclic Redundancy Check) values, which are special mathematical operations designed for error checking and are frequently referred to in serial communication protocols.

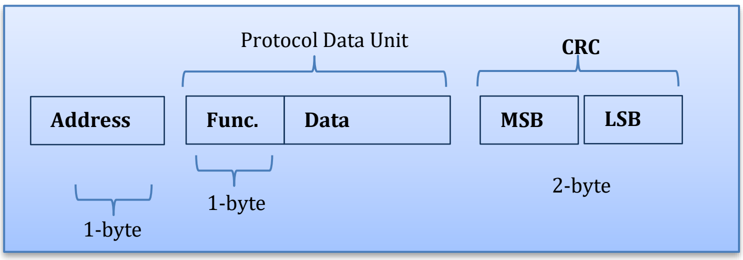


**Figure 2: Modbus Data Frame**

## Restrictions

In addition to general Modbus restrictions, there are different restrictions on device communication. These restrictions are important for communication and healthy operation of the device.

### *Modbus Restrictions*

Modbus protocols can only be realized the communication between devices under certain conditions. For example, if the message is send to the device with a wrong address by a master device, it cannot be returned.

In addition, Modbus also includes serial communication restrictions, so it can send and received up to 256 bytes data frame. In this particular device, up to 60 bytes message frames can be carried.

**CRC (Cyclic Redundancy Check)**

The last two bytes of each message is reserved for CRC. The control is done by these two bytes whether the message block is damaged and changed for any reason.

The CRC is valid used method of fault control method for digital communication and is also valid method for Modbus. If a data frame with fault is send, a fault message will be returned.

**Wrong and Undefined Functions**

Even if the wrong and undefined functions are routed to the correct device address, no action will be taken and an error will be returned. Therefore, it is important to use defined functions in the specified way.

**Infraction of the Read-Only Feature**

If it is tried to write a value to the only-read data structure, this message will be interpreted but the value will not be written. In such a case, there will be a fault return due to the Modbus protocol.

### *Earth Fault & Short Circuit Indicator Restrictions*

In addition to device protocol dependencies, there are registers that are subject to some restrictions and special writing rules due to capacity of device, usage safety and structural safety.

ESI allows 60 bytes of data to be exchanged. This restriction has been made so that the number of registers is low and communication can run smoothly.

The device can provide 4 baud rates (4800, 9600, 19200 and 38400).

Parity option can define with 3 different value (odd, even, none) for communication.

Data bits are predefined and 8-bits.

## Writable Registers

12 writable registers in the device are defined in the below table.

These registers have both reading and writing features. They hold different values from device addresses to temperature protection. All of them have 16-bit digital data type.

While function 03 is used to read writable registers, functions 06 and 16, which are write functions, can also be used to change the values of these registers. However, it should not be forgotten that there are areas to influence the communication directly when their values change. For example, if we change the value of address, we must write newly saved value on the device address otherwise it should not be forgotten that it cannot provide communication. To avoid this and similar problems, it is needed to be sure of the values to be written before the write operation.

| Address | Register Name | Interval / Step |
|---|---|---|

| | | |
|---|---|---|
| 1 | PI_SET (A) | 0-1000 (A) / 50 |
| 2 | tPI_SET (ms) | 0-800 / 40 |
| 3 | IE_SET (A) | 0-200 / 20 |
| 4 | tIE_SET (ms) | 0-800 / 40 |
| 5 | tRESET_SET (min) | 0-240 / 30 |
| 6 | Hour | 0-23 / 1 |
| 7 | Minute | 0-59 / 1 |
| 8 | Second | 0-59 / 1 |
| 9 | Day | 0-31 / 1 |
| 10 | Month | 0-12 / 1 |
| 11 | Year | 0-99 / 1 |
| 12 | Address | 0-254 / 1 |
| 13 | Baud Speed | See Explanation 1 |
| 14 | Parity | See Explanation 1 |
| 15 | Reset Fault | See Explanation 2 |
| 16 | Clear Events | See Explanation 2 |

**Table 2: Writable Registers**

**Explanation 1**: These registers cannot be written in simple form. For example, the Baud register, which holds data communication speed information, can only take certain values, these values are specified in a set, and the user selects these values from within that set. Detailed information is described under "Special Register Values".

If baud rate or parity values are changed, it is needed to re-adjust the baud rate and parity values of master device to communicate with the slave device again.

**Explanation 2**: These registers are created to reset the fault or clear the events via serial port. Only value “1” can be written to these registers.

In order to reset faults, it must be written value “1” to the register “Reset Fault” via serial port.

In order to clear events, it must be written value “1” to the register “Clear Events”.

### *Special Register Value*

**Baudrate Register:** 4800, 9600, 19200, 38400 kbps options are adjusted according to values explained below.

If the Baud register is 1, the baudrate will be 4800, 2 refers to 9600 kbps, 3 refers to 19200 kbps, 4 refers to 38400 kbps. If a value greater than 4 is written to the baud register, the mode of value is taken according to the number 4.

**Stop Bit: and Parity:** 8N1(none parity and 1 stop bit), 8E1 (even parity and 1 stop bit), 8O1 (odd parity and 1 stop bit), 8O2 (odd parity and 2 stop bit) options are adjusted according to values explained below.

If the stop bit and parity register is 1, the stop bit and parity will be 8N1, 2 refers to 8E1, 3 refers to 8O1 and 4 refers to 8O2.

### *Read Function*

Function 03 is the read function for the writable registers.

**Example:**

If the device address is 4 and reading total of 2 registers starting from 5.

**Sent Message:** **04 03 00 05 00 02 D4 5F**

**04 –** Device Address

**03 –** Function Code

**0005 –** The first register address desired to read

**0002 –** The amount of the register desired to read

**D45F –** CRC fault control bytes

**Received Message:** **04 03 04 00 5A 00 02 0E E1**

**04 –** Device Address

**03 –** Function Code

**04 –** Number of data byte, it gives information about holding data the next 4 bytes.

**005A –** The first register desired to read (0005 – ***tReset****: automatic reset time*) value

(005A = 90)

**0002 –** The second register desired to read (0006 – ***Time****: Device Time*) value (0002 = 2)

**0EE1 –** CRC fault control bytes

### *Writing Functions*

Two different writing functions are defined before for writing registers in Modbus. These functions are Function 06 and Function 16. While Function 06 allows writing on only one register, Function 16 can provide writing on multiple registers.

**Function 06 example:**

Let the device address is 04 and 3rd Register that is IE_SET(earth current fault limit) value is 40.

**Sent message:** **04 06 00 03 00 28 79 81**

**04 –** Device Address

**06 –** Function Code

**0003 –** Register address

**0028 –** Writing value

**7981 –** CRC fault control bytes

**Received message:** **04 06 00 03 00 28 79 81** if writing operation is done without any problem, the sent message and received message will be same. If there is a problem a fault message will return.

**Function 16 example:** This function allows multiple writes for successive registers.

Let's update the device date setting, device address is 04 and value of 9th register is 25, value of 10th register is 05 and value of 11th register is 17.

**Sent message:** **04 10 00 09 00 03 06 00 19 00 05 00 11 f7 65** The message length will vary depending on the number of registers desired to be written. According to the example, we send a 15-byte data block.

**04 –** Device Address

**10 –** Function Code (16)

**0009 –** Starting address

**0003 –** Amount of register

**06 –** Number of data byte, it will hold the data that will write on the registers followed 6 bytes respectively.

**0019 –** (**0009**) The value to be written to register at the first address specified.

**0005 –** (**000A**) The value to be written to register at the second address specified.

**0011 –** (**000B**) The value to be written to register at the third address specified.

**F765 –** CRC fault control bytes

**Returned Message:** **04 10 00 09 00 03 50 5F** – if message is successful

**04 –** Device Address

**10 –** Function Code

**0009 –** Starting address

**0003 –** Amount of written register

**505F –** CRC fault control bytes

If the write operation fails in the write functions, a fault message of 5 bytes will be returned.

## Analog Registers

There are 164 analog input registers that are defined in the table below. These registers hold only readable input values. The registers holding different qualifiers in each 32-bit length are mapped in 4-byte fields with "Big-Endian" notations.

Two registers must be read together for the exact reading of any of the analog registers. Only a high or only low value of a register can be read if needed.

| Address | Register Name | Explanation |
|---|---|---|
| 1 | A Phase Current (A) | 0 - 1000/ 1 |
| 2 | B Phase Current (A) | 0 - 1000/ 1 |
| 3 | C Phase Current (A) | 0 - 1000/ 1 |
| 4 | Earth Current (A) | 0 - 1000/ 1 |
| 5 | 1. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 6 | 1. Event Current Information | 0-1000 / 1 |
| 7 | 1. Event Time | |
| 8 | 1. Event Minute | |
| 9 | 1. Event Second | |
| 10 | 1. Event Day | |
| 11 | 1. Event Month | |
| 12 | 1. Event Year | |
| 13 | 2. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 14 | 2. Event Current Information | 0-1000 / 1 |
| 15 | 2. Event Hour | |
| 16 | 2. Event Minute | |
| 17 | 2. Event Second | |
| 18 | 2. Event Day | |
| 19 | 2. Event Month | |
| 20 | 2. Event Year | |
| 21 | 3. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 22 | 3. Event Current Information | 0-1000 / 1 |
| 23 | 3. Event Hour | |
| 24 | 3. Event Minute | |
| 25 | 3. Event Second | |
| 26 | 3. Event Day | |
| 27 | 3. Event Month | |

| | | |
|---|---|---|
| 28 | 3. Event Year | |
| 29 | 4. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 30 | 4. Event Current Information | 0-1000 / 1 |
| 31 | 4. Event Hour | |
| 32 | 4. Event Minute | |
| 33 | 4. Event Second | |
| 34 | 4. Event Day | |
| 35 | 4. Event Month | |
| 36 | 4. Event Year | |
| 37 | 5. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 38 | 5. Event Current Information | 0-1000 / 1 |
| 39 | 5. Event Hour | |
| 40 | 5. Event Minute | |
| 41 | 5. Event Second | |
| 42 | 5. Event Day | |
| 43 | 5. Event Month | |
| 44 | 5. Event Year | |
| 45 | 6. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 46 | 6. Event Current Information | 0-1000 / 1 |
| 47 | 6. Event Hour | |
| 48 | 6. Event Minute | |
| 49 | 6. Event Second | |
| 50 | 6. Event Day | |
| 51 | 6. Event Month | |
| 52 | 6. Event Year | |
| 53 | 7. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 54 | 7. Event Current Information | 0-1000 / 1 |
| 55 | 7. Event Hour | |
| 56 | 7. Event Minute | |
| 57 | 7. Event Second | |
| 58 | 7. Event Day | |
| 59 | 7. Event Month | |
| 60 | 7. Event Year | |
| 61 | 8. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 62 | 8. Event Current Information | 0-1000 / 1 |
| 63 | 8. Event Hour | |
| 64 | 8. Event Minute | |
| 65 | 8. Event Second | |
| 66 | 8. Event Day | |
| 67 | 8. Event Month | |
| 68 | 8. Event Year | |
| 69 | 9. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase |

| | | |
|---|---|---|
| | | 4: Earth Current |
| 70 | 9. Event Current Information | 0-1000 / 1 |
| 71 | 9. Event Hour | |
| 72 | 9. Event Minute | |
| 73 | 9. Event Second | |
| 74 | 9. Event Day | |
| 75 | 9. Event Month | |
| 76 | 9. Event Year | |
| 77 | 10. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 78 | 10. Event Current Information | 0-1000 / 1 |
| 79 | 10. Event Hour | |
| 80 | 10. Event Minute | |
| 81 | 10. Event Second | |
| 82 | 10. Event Day | |
| 83 | 10. Event Month | |
| 84 | 10. Event Year | |
| 85 | 11. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 86 | 11. Event Current Information | 0-1000 / 1 |
| 87 | 11. Event Hour | |
| 88 | 11. Event Minute | |
| 89 | 11. Event Second | |
| 90 | 11. Event Day | |
| 91 | 11. Event Month | |
| 92 | 11. Event Year | |
| 93 | 12. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 94 | 12. Event Current Information | 0-1000 / 1 |
| 95 | 12. Event Hour | |
| 96 | 12. Event Minute | |
| 97 | 12. Event Second | |
| 98 | 12. Event Day | |
| 99 | 12. Event Month | |
| 100 | 12. Event Year | |
| 101 | 13. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 102 | 13. Event Current Information | 0-1000 / 1 |
| 103 | 13. Event Hour | |
| 104 | 13. Event Minute | |
| 105 | 13. Event Second | |
| 106 | 13. Event Day | |
| 107 | 13. Event Month | |
| 108 | 13. Event Year | |
| 109 | 14. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 110 | 14. Event Current Information | 0-1000 / 1 |

| | | |
|---|---|---|
| 111 | 14. Event Hour | |
| 112 | 14. Event Minute | |
| 113 | 14. Event Second | |
| 114 | 14. Event Day | |
| 115 | 14. Event Month | |
| 116 | 14. Event Year | |
| 117 | 15. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 118 | 15. Event Current Information | 0-1000 / 1 |
| 119 | 15. Event Hour | |
| 120 | 15. Event Minute | |
| 121 | 15. Event Second | |
| 122 | 15. Event Day | |
| 123 | 15. Event Month | |
| 124 | 15. Event Year | |
| 125 | 16. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 126 | 16. Event Current Information | 0-1000 / 1 |
| 127 | 16. Event Hour | |
| 128 | 16. Event Minute | |
| 129 | 16. Event Second | |
| 130 | 16. Event Day | |
| 131 | 16. Event Month | |
| 132 | 16. Event Year | |
| 133 | 17. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 134 | 17. Event Current Information | 0-1000 / 1 |
| 135 | 17. Event Hour | |
| 136 | 17. Event Minute | |
| 137 | 17. Event Second | |
| 138 | 17. Event Day | |
| 139 | 17. Event Month | |
| 140 | 17. Event Year | |
| 141 | 18. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 142 | 18. Event Current Information | 0-1000 / 1 |
| 143 | 18. Event Hour | |
| 144 | 18. Event Minute | |
| 145 | 18. Event Second | |
| 146 | 18. Event Day | |
| 147 | 18. Event Month | |
| 148 | 18. Event Year | |
| 149 | 19. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 150 | 19. Event Current Information | 0-1000 / 1 |
| 151 | 19. Event Hour | |
| 152 | 19. Event Minute | |

| | | |
|---|---|---|
| 153 | 19. Event Second | |
| 154 | 19. Event Day | |
| 155 | 19. Event Month | |
| 156 | 19. Event Year | |
| 157 | 20. Event Phase Information | 1: A Phase 2: B Phase 3: C Phase 4: Earth Current |
| 158 | 20. Event Current Information | 0-1000 / 1 |
| 159 | 20. Event Hour | |
| 160 | 20. Event Minute | |
| 161 | 20. Event Second | |
| 162 | 20. Event Day | |
| 163 | 20. Event Month | |
| 164 | 20. Event Year | |

**Table 3: Analog Registers**

### *Reading Function*

Reading function of analog registers is 04.

**Example:** Let assume that the device address is defined as “4” and it is desired to read current values. The starting address will be “1” and it means that it is desired to read 4 register in total.

**Sent message:** **04 04 00 01 00 04 A0 5C**

**04 –** Device Address

**04 –** Function Code

**0001 –** Starting address

**0004 –** Amount of register desired to read.

**A05C –** CRC fault control byte

**Returned message:** **04 04 08 00 03 00 02 00 03 00 00 8F 01**

**04 –** Device Address

**04 –** Function Code

**08 –** Number of data byte

**0003 –** (**0001**) value of first register

**0002 –** (**0002**) value of second register

**0003 – (0003)** value of third register

**0000 – (0004)** value of fourth register

**8F01 –** CRC fault control byte

# INDEX