# Introductions

## What is Modbus®?

Introduction to Function Codes

Introduction to Data Types

Register Addressing

Comm Settings and Modbus® Address

## Addresses and Values

Address Map

## Appendix: Reading and Writing

Data Types

Function Codes

Creating Modbus® Commands

Programming Examples

Modbus® Responses
What is Modbus®?

Modbus® is an industry standard serial protocol for communication between industrial electronic devices and various control and display equipment, such as PLCs, SCADA systems, and panel meters. Modbus® was developed by Modicon® and released for use in 1979. It is an open protocol, available for use without royalties. It is typically used to set parameters or gather data from various instrumentation.

The official Modbus® specification can be found at: [www.modbus.org/specs.php](http://www.modbus.org/specs.php)

A good source for an introduction to Modbus® is: [www.simplymodbus.ca/FAQ.htm#Modbus](http://www.simplymodbus.ca/FAQ.htm#Modbus)

Modbus® on Seametrics iMAG 4700 and AG3000 Magmeters

Several iMAG 4700 and AG3000 models can be factory configured for Modbus® RTU communication, providing a half-duplex, isolated, RS485 serial communications port using the Modbus® messaging protocol. The interface contains an RS485 transceiver which implements a fail-safe receiver to assure that a mark condition will be sensed even when the cable is disconnected or undriven. Under this condition, a bias network is unnecessary for the iMAG 4700/AG3000 interface to function properly. It is assumed, however, that the master contains either a Fail Safe receiver or a bias network for proper operation of the RS485 network as whole. A useful reference on this subject is an Applications Note from Texas Instruments: [www.ti.com/lit/an/slyt514/slyt514.pdf](http://www.ti.com/lit/an/slyt514/slyt514.pdf)

For attachment to the RS485 network, the iMAG 4700/AG3000 comes equipped with a 6 or 8 conductor cable. For Modbus®, the following connections are used:

<table>
<thead>
<tr>
<th>Ground</th>
<th>White wire</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/TX[+]</td>
<td>Orange wire</td>
</tr>
<tr>
<td>B/RX[-]</td>
<td>Blue wire</td>
</tr>
</tbody>
</table>

The iMAG 4700/AG3000 Modbus® interface can work with network cable lengths up to 50 feet without termination. If a longer cable length is used, then DC or AC termination may be needed if the meter is at the end of the network. Under this condition consult the factory for further details if this option is needed.

Introduction to Function Codes

The Modbus® function codes implemented for the iMAG 4700/AG3000 consist of the following:

<table>
<thead>
<tr>
<th>Function code Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>Read Holding Register</td>
</tr>
<tr>
<td>0x04</td>
<td>Read Input Register</td>
</tr>
<tr>
<td>0x06</td>
<td>Write Single Register</td>
</tr>
<tr>
<td>0x10</td>
<td>Write Multiple Registers</td>
</tr>
</tbody>
</table>

For more details, see the appendix.

⚠️ When quantities are written to the meter, they are also stored in an internal EEPROM device. As such, there is a limit as to how many times the device may be written. With this in mind, the write operation should be used cautiously and sparingly.
Introduction to Data Types

Data and parameters are stored in 16-bit registers (1 word), each register being 2 bytes.

The following data types are used in the iMAG 4700/AG3000 Modbus® interface:

- **Integer**: 16 bit unsigned value corresponding to 1 register
- **Float**: 32 bit IEEE floating point number corresponding to 2 sequential registers
- **String**: 16 ASCII characters correspond to 16 sequential registers

For more details, see the appendix.

Register Addressing

The physical register addresses for the iMAG 4700/AG3000 meters start numbering from zero (0-based addressing)—the first address is 0, the second is 1, etc. On the other hand, Modbus® protocol considers the first logical address to be 1, the second logical address to be 2, etc. (1-based addressing). For example, to view or set TUNITS, you have to read the physical address 1000 (0x03E8). Some programs and equipment when asked to read address 1000 (0x03E8) will read that physical address. Others however will read the logical address, which is actually the physical address 999 (0x03E7). With these programs and equipment you must add a one to the address—thus in this example you would request TUNITS at address 1001 (0x03E9).

Still other programs and equipment require the addition of such numbers as 40,000 or 400,000 to the address to indicate reading/writing to holding registers. These may or may not require the addition of one to the physical address, as well. Check with your system documentation to determine what style of register addressing is required.

All addressing in this document is based on the physical address (0-based addressing).

If you have trouble establishing communication, pick a single address such as TOTAL-FWD and poll the 0-based address. Then, poll that address plus 1 or that address -1.

Comm Settings and Modbus® Address

**COMM Settings**

When communicating between a controller and the meter, both the controller and the meter must be set to the same baud rate and parity. These values must be set using the on-meter menu system.

To access the COMM settings: From the main menu navigate to the EXIT tab and tap five times. This will bring you to a submenu. Navigate to the COMM tab and select your desired communication setting. The default is 19,200 baud with no parity (19200 NONE). See the COMM entry in the Address Map table on the following pages for other options.

**Modbus® Slave ID**

Each slave device must have a unique Modbus® slave ID, which the controller will use to communicate with that particular device. The iMAG 4700/AG3000 meters default to address 1. If you need to change this, you can do so using the on-meter menu system.

To access the Modbus® slave ID setting (MBID): From the main menu navigate to the EXIT tab and tap five times. This will bring you to a submenu. Navigate to the MBID tab and set your desired address (1–247).
### Address Map

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>On Menu or Display</th>
<th>Via Modbus</th>
<th>R/W</th>
<th>Data Values</th>
<th>Address (0-Based)</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TUNIT</strong></td>
<td>View or select units for displaying the contents of the totalizer</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0 Gallons 1 Gallons x 10 2 Gallons x 100 3 Gallons x 1000 4 Million Gallons 5 Cubic Feet 6 Cubic Feet x 1000 7 Second Foot Day 8 Million Cubic Feet 9 Imperial Gallons 10 Imp. Gal. x 1000 11 Million Imp. Gal. 12 Liters 13 Kilo Liters 14 Mega Liters 15 Cubic Meters 16 Cubic Meters x 1000 17 Barrels 18 Acre Inches 19 Acre Feet 20 Fluid Ounces</td>
<td>1000</td>
<td>0x3E8</td>
</tr>
<tr>
<td><strong>RUNIT</strong></td>
<td>View or select units for displaying the current computed rate. (Need also to set a time base, RUNIT TIME, below.)</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0 Gallons 1 Million Gallons 2 Cubic Feet 3 Imperial Gallons 4 Million Imp. Gal. 5 Liters 6 Mega Liters 7 Cubic Meters 8 Barrels 9 Fluid Ounces</td>
<td>1001</td>
<td>0x3E9</td>
</tr>
<tr>
<td><strong>RUNIT TIME</strong></td>
<td>View or set time base for RUNITs</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0 Second 1 Minute 2 Hour 3 Day</td>
<td>1002</td>
<td>0x3EA</td>
</tr>
<tr>
<td><strong>DAMP</strong></td>
<td>View or select the amount of damping so as to stabilize the display of the rate. The selection corresponds to a window, in cycles, over which averaging takes place. These windows, in turn, are overlapped in time such that a new average value is computed every cycle.</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0-99 cycles</td>
<td>1003</td>
<td>0x3EB</td>
</tr>
<tr>
<td><strong>SETF</strong></td>
<td>View or set the maximum frequency available at the high frequency output. This frequency corresponds to Q4, the maximum flow rate for a given pipe diameter. The frequency output is scaled between zero flow and Q4.</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0 50 Hz 1 kHz 2 2 kHz 3 5 kHz 4 10 kHz</td>
<td>1004</td>
<td>0x3EC</td>
</tr>
<tr>
<td><strong>SETP UNITS</strong></td>
<td>View or select units for pulse output</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>0 Gallons 1 Cubic Feet 2 Imperial Gallons 3 Liters 4 Cubic Meters 5 Barrels 6 Fluid Ounces</td>
<td>1005</td>
<td>0x3ED</td>
</tr>
<tr>
<td><strong>STATUS</strong></td>
<td>Reports communication failure and empty pipe</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Data Bit</td>
<td>Data Selection</td>
<td>5000</td>
</tr>
<tr>
<td><strong>SET4</strong></td>
<td>View or set the rate corresponding to the 4 mA setting for the 4-20 mA loop. It specifies the lower limit of the rate that will result in controlling 4 mA in the loop.</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>00000.0 to 99999.9 (units specified by RUNIT)</td>
<td>7000</td>
<td>0x1B58</td>
</tr>
</tbody>
</table>

*Continued on next page*
### Address Map (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>On Menu or Display</th>
<th>Via Modbus</th>
<th>R/W</th>
<th>Data Values</th>
<th>Address</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET20</td>
<td>View or set the rate corresponding to the 20 mA setting for the 4-20 mA loop. It specifies the upper limit of the rate that will result in controlling 20 mA in the loop. The result of defining the SET4 and SET20 limits is to scale the rate between these limits to the span between 4 mA and 20 mA</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>00000.0 to 99999.9 (units specified by RUNIT)</td>
<td>7002</td>
<td>0x1B5A</td>
</tr>
<tr>
<td>SETP</td>
<td>View or set the amount of fluid measured, in the selected units, for emitting a pulse output. As an example, if the rate is 1 gallon/sec, and SETP is 10 gallons, then one pulse is emitted for every 10 gallons which are metered. This occurs once every 10 seconds. Note, also, that the units for SETP are independently set.</td>
<td>Y</td>
<td>Y</td>
<td>R/W</td>
<td>00000.0 to 99999.9</td>
<td>7004</td>
<td>0x1B5C</td>
</tr>
<tr>
<td>FLOW RATE</td>
<td>View flow rate in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Computed rate in units specified by RUNIT</td>
<td>7006</td>
<td>0x1B5E</td>
</tr>
<tr>
<td>TOTAL - FWD</td>
<td>View forward flow total in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Current total volume in forward direction in unit specified by TUNIT</td>
<td>7008</td>
<td>0x1B60</td>
</tr>
<tr>
<td>TOTAL - REV</td>
<td>View reverse flow total in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Current total volume in reverse direction in unit specified by TUNIT</td>
<td>7010</td>
<td>0x1B62</td>
</tr>
<tr>
<td>ZRADC</td>
<td>One of two calibration constants</td>
<td>N</td>
<td>Y</td>
<td>R</td>
<td>Calibrated constant derived from calibration procedure</td>
<td>7012</td>
<td>0x1B64</td>
</tr>
<tr>
<td>FSADC</td>
<td>One of two calibration constants</td>
<td>N</td>
<td>Y</td>
<td>R</td>
<td>Calibrated constant derived from calibration procedure</td>
<td>7014</td>
<td>0x1B66</td>
</tr>
<tr>
<td>CUTOFF</td>
<td>Corresponds to the lowest measurable flow rate</td>
<td>N</td>
<td>Y</td>
<td>R</td>
<td>Lowest measurable flow rate in gpm</td>
<td>7016</td>
<td>0x1B68</td>
</tr>
<tr>
<td>Q4</td>
<td>Corresponds to the maximum flow rate</td>
<td>N</td>
<td>Y</td>
<td>R</td>
<td>Maximum measurable flow rate in gpm based on flow rate of 10 meters/sec and a given pipe diameter</td>
<td>7018</td>
<td>0x1B6A</td>
</tr>
<tr>
<td>VOLTAGE</td>
<td>Reports external voltage used to power meter</td>
<td>N</td>
<td>Y</td>
<td>R</td>
<td>Measured external voltage</td>
<td>7020</td>
<td>0x1B6C</td>
</tr>
<tr>
<td>SERIAL #</td>
<td>View meter serial number</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>ASCII string containing serial number</td>
<td>6000</td>
<td>0x1770</td>
</tr>
<tr>
<td>MODEL #</td>
<td>View meter model number</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>ASCII string containing model number</td>
<td>6008</td>
<td>0x1778</td>
</tr>
<tr>
<td>LOWER FIRMWARE</td>
<td>View body firmware version number</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>ASCII string containing firmware version</td>
<td>6016</td>
<td>0x1780</td>
</tr>
<tr>
<td>UPPER FIRMWARE</td>
<td>View head firmware version number</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>ASCII string containing firmware version</td>
<td>6024</td>
<td>0x1788</td>
</tr>
<tr>
<td>MBID</td>
<td>View or set the Modbus slave ID</td>
<td>Y</td>
<td>N</td>
<td>R/W</td>
<td>001 to 247 Default is 001</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>COMM</td>
<td>View or set communication parameters</td>
<td>Y</td>
<td>N</td>
<td>R/W</td>
<td>Baud Rate Parity 38400 None 38400 Even 38400 Odd 19200 None 19200 Even 19200 Odd 9600 None 9600 Even 9600 Odd Default is 19200/None</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Note: Due to values added in newer models, your address map may be slightly different.
APPENDIX: READING AND WRITING

Data Types
Data and parameters are stored in 16-bit registers (1 word), each register being 2 bytes. All data is in “big-endian” format, or high word/high byte first. This is also sometimes called “float-inverse” or “Float AB CD”. The symptom of using the wrong format is a response value that is readable but is entirely wrong.

The following data types are used in the iMAG 4700/AG3000 Modbus® interface:

- **Integer**: 16 bit unsigned value corresponding to 1 register
- **Float**: 32 bit IEEE floating point number corresponding to 2 sequential registers
- **String**: 16 ASCII characters correspond to 16 sequential registers

Function Codes
The Modbus® function codes implemented for the iMAG 4700/AG3000 consist of the following:

<table>
<thead>
<tr>
<th>Function code</th>
<th>Description</th>
</tr>
</thead>
</table>
| **0x03** 03   | Read Holding Register  
Can be used to read one or more integer, float, or string values. Requires starting register address and number of registers to read. |
| **0x04** 04   | Read Input Register  
Can be used to read one or more integer, float, or string values. Requires starting register address and number of registers to read. |
| **0x06** 06   | Write Single Register  
Can be used to write one integer value. Requires register address and value. |
| **0x10** 16   | Write Multiple Registers  
Can be used to write one or more integer, float, or string values. Requires starting register address, number of registers to write, and the values. |

**Remember!** Values may take more than one register. For example, if reading or writing two float values, you would specify four registers, as each floating point number corresponds to two sequential registers.

Inhanced Accuracy of Very Large Totals
The values for FLOWRATE, FORWARD TOTAL, and REVERSE TOTAL are stored internally in the meter as 64-bit, double precision IEEE numbers. Because the Modbus float values stated previously are represented by 32-bit IEEE single precision floating point numbers, when a total exceeding 16,777,216 units is transmitted by Modbus, the 32-bit float value will include a certain amount of rounding which is relatively insignificant compared to the value itself.

If absolute precision is required in the TOTAL values communicated by Modbus, the FLOWRATE, FORWARD TOTAL and REVERSE TOTAL can be expressed as 64-bit IEEE (double) floating point numbers each corresponding to 4 sequential registers (8 bytes.)

These additional 64-bit addresses are:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>On Menu or Display</th>
<th>Via Modbus</th>
<th>R/W</th>
<th>Data Values</th>
<th>Address</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW RATE</td>
<td>View flow rate in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Computed rate in units specified by RUNIT</td>
<td>3000</td>
<td>Double Float</td>
</tr>
<tr>
<td>TOTAL – FWD</td>
<td>View forward flow total in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Current total volume in forward direction in unit specified by TUNIT</td>
<td>3004</td>
<td>Double Float</td>
</tr>
<tr>
<td>TOTAL – REV</td>
<td>View reverse flow total in selected units</td>
<td>Y</td>
<td>Y</td>
<td>R</td>
<td>Current total volume in reverse direction in unit specified by TUNIT</td>
<td>3008</td>
<td>Double Float</td>
</tr>
</tbody>
</table>

Each quantity occupies 4 registers.
Creating a Modbus® Command

When using your controller (PLC, SCADA system, etc.) to access the meter, you will typically be asked for the following:

For Reading: (function code 0x03 or 0x04/03 or 04 decimal)
- Modbus slave ID
- Function code\(^1\)
- Starting register address\(^1\)
- Number of registers to read\(^2\)

For Writing—single integer: (function code 0x06/06 decimal)
- Modbus slave ID
- Function code\(^1\)
- Register address\(^1\)
- Value to write\(^3\)

For Writing—multiple values: (function code 0x10/16 decimal)
- Modbus slave ID
- Function code\(^1\)
- Starting register address\(^1\)
- Quantity of registers\(^2\)
- Byte count\(^2\)
- Values to write\(^3\)

From the information you enter, the controller will construct a command string to send to the meter. The controller will automatically append a CRC code at the end. This is a two-byte computed value unique to that string. When the meter receives the command string, it will compute its own CRC and verify that it matches the one sent by the controller, thus verifying that all bytes were received correctly. If the CRC codes do not match, an error will be returned to the controller.

---

1 Some systems may combine the function code and register address. For example, 41001 may mean read holding register 1001, same as function 03 with a register address of 1001 (0x03E9). Consult your controller documentation for details specific to your equipment.

2 Some systems will ask for the number of values and type of value rather than quantity of registers. It will automatically compute the number of registers and byte count. For example, the user would specify 2 values of type float and the system would translate that to 4 registers (or 8 bytes).

3 Some systems may ask for values in decimal, others may ask for values in hex.
Programming Examples

In this section, some typical programming examples are given for programming the iMAG 4700/AG3000 meter through the Modbus® interface. These examples display the final strings that would be sent to the meter. Remember that in most cases, you will enter requested values into the particular interface of your controller, which will then create the actual programming strings that are sent to the meter. (Note that a space has been added between each byte for legibility. These spaces would not be in an actual command string.)

**To set TUNIT** (Total Units)

To program the total units for liters, we need to write the TUNIT register at address 1000 (0x03E8) with a value of 12 (0x0C). To accomplish this, we can use the Write Single Register command (0x06) with the following string.

```
01 06 03 E8 00 0C 09 BF
```

**To set RUNIT** (Rate Units)

To program the rate units for liters, we need to write the RUNIT register at address 1001 (0x03E9) with a value of 5. To accomplish this, we can use the Write Single Register command (0x06) with the following string.

```
01 06 03 E9 00 05 98 79
```

**To set RUNIT Time** (Rate Unit Time Base)

To program the rate units time base for hours, we need to write the RUNIT TIME register at address 1002 (0x03EA) with a value of 0. To accomplish this, we can use the Write Single Register command (0x06) with the following string.

```
01 06 03 EA 00 00 A8 7A
```
To set TUNIT, RUNIT, and RUNIT Time with a single command

As an alternative to writing each register individually with the Write Single Register command, the Write Multiple Registers command (0x10) can be used to write all 3 registers in one command operation. The following string can be used to write TUNIT, RUNIT, and RUNIT TIME registers. In this example, the same three values are written as were written in the previous examples, i.e., total units in liters, rate units in liters, and time base in seconds.

$$\begin{array}{c}
01 & 10 & 03 & E8 & 00 & 03 & 06 & 00 & 0C & 00 & 05 & 00 & 00 & 6C & 78 \\
MBID & Function code & Starting register addr & Number of registers & Byte Count & Value for first register & Value for second register & Value for third register & CRC
\end{array}$$

To Monitor Flow Rate and Total

Modbus® can be used to monitor the flow rate and total from the iMAG 4700/AG3000 meter. This can be accomplished with either the Read Holding Registers (0x03) or Read Input Registers (0x04) command. To read flow rate, flow total-forward, and flow-total-reverse in one operation we would use a starting address of 7006 (0x1B5E) and the quantity of registers of 6. Six registers are required since these are floating point numbers requiring 2 registers for each value. Either function code 0x03 or 0x04 can be used.

$$\begin{array}{c}
01 & 04 & 1B & 5E & 00 & 06 & 17 & 3E \\
MBID & Function code & Starting register addr & Number of registers & CRC
\end{array}$$

Modbus Board Jumper Configuration (Meter is shipped with jumpers set for RS485 communication by default)
Modbus® Responses

Following transmission of a Modbus® command, the meter will either return a successful response or an error response.

Successful Responses

Successful responses will return the following:

- **Read**: (0x03 or 0x04) Function code, byte count, requested value(s)
- **Write Single Reg**: (0x06) Function code, register address, value written
- **Write Multiple Reg**: (0x10) Function code, starting register, quantity of registers written

Failed Responses

Failed responses will return the following:

- **Read**: (0x03 or 0x04) Error code 0x83 or 0x84, exception code
- **Write Single Reg**: (0x06) Error code 0x86, exception code
- **Write Multiple Reg**: (0x10) Error code 0x90, exception code

Exception code 4

An exception code 4 will be returned in the following circumstances:

- **Invalid Address**: An attempt has been made to write to a non-existent register.
- **Wrong number of registers**: The data type implies that a certain number of registers need to designated either for read or write operations. For example, if we wish to write a floating point value then the number of designated registers must be modulus 2 since 2 registers are needed to hold a floating point value. As an example, if a single floating point number is read, and only 1 register is specified for the quantity of registers, then an exception code is sent.
- **Addressed registers contain mixed data types**: When multiple registers are either written or read, all the implied data types must be must be the same, e.g, all floating point or all integers. If there is a mixture, then an exception code is sent.

Timeout

The conditions under which a timeout results is as follows:

- **Wrong slave ID**: An incorrect slave ID is designated.
- **Wrong COMM parameters**: The baud rate and parity are incorrectly selected.
- **Wrong register address**