VERIS H8035 and H8036

MODBUS IMPLEMENTATION SPECIFICATION

OVERVIEW

This document describes the implementation of Modbus protocol used in the Veris H8035 and H8036 power meters. It is intended to assist control system programmers who may need to add support to their systems to communicate with these power meters. An example program that runs in Windows 95 and its C source code are available to illustrate the Modbus implementation. Additional information about Modbus itself available from Modicon in the "Modicon Modbus Protocol Reference Guide", PI-MBUS-300, Rev J, which is available on the Internet at: http://www.modicon.com/techpubs/toc7.html

Please contact support at Veris Industries for additional information, 503-670-1277 or 800-354-8556, or email support@veris.com.

COMMUNICATION PARAMETERS

Parameter	Value
Modbus Framing	RTU (binary)
Baud Rate	9600
Data Bits	8
Parity	none
Stop Bits	1

The power meter operates as a slave device. It expects a Modbus master device to transmit queries, which it will answer.

MODBUS COMMAND QUICK REFERENCE

03 - Read Holding Register (8 bytes)

ADDR	**03**h	**FIRST** MSB	**FIRST** LSB	**NUM** MSB	**NUM** LSB	**CRC** LSB	**CRC** MSB

ADDR = which meter to read, **FIRST** = first point to read, **NUM** = number of points to read

06 - Preset Single Register (8 bytes)

ADDR	**06**h	**POINT** MSB	**POINT** LSB	**DATA** MSB	**DATA** LSB	**CRC** LSB	**CRC** MSB

ADDR = which meter to write, **POINT** = point number to write, **DATA** = value to store

17 - Report Slave ID (4 bytes)

ADDR	**11**h	**CRC** LSB	**CRC** MSB

ADDR = which meter to respond

DATA FORMAT

All data is available in two formats, 16 bit integers and 32 bit floating point numbers. All data points are available in both formats, see "Modbus Register Addressing" for details about the addresses allocated for each group.

For systems which can process floating point numbers, it is suggested that the float values be used because no scaling is required. This can save considerable set-up effort and lower the total installation cost because the numbers represent the true value of each point, regardless of the type of data or the product's amperage range. Using floating point numbers eliminates the need for multipliers to obtain correct data.

For systems which can not handle floating point numbers, integer values may be used. Each of these must be multiplied by a "multiplier" to obtain the correct data. Many data values require a different multiplier depending on the product's maximum amperage range. See "Using Integer Data Types" for details about the required multiplier values. In some cases it may be desirable to use integer values because they consume only 16 bits instead of 32, theoretically allowing them to be transmitted twice as fast. In practice, less speed increase can be realized, due to the Modbus protocol overhead, including the required silent times for message framing.

MODBUS REGISTER ADDRESSING

This table lists the addressed assigned to each data point. For floating point format variables, each data point appears twice because two 16-bit addresses are required to hold a 32-bit float value.

for H8035 products

Modbus Addr	Typical Offset	Units	Description	Data Type
40001	+0	KWH	Energy Consumption, LSW	Integer (mult req'd)
40002	+1	KWH	Energy Consumption, MSW	Integer (mult req'd)
40003	+2	KW	Demand (power)	Integer (mult req'd)
40257	---	KWH	Energy Consumption	Float, upper 16 bits
40258		KWH	Energy Consumption	Float, lower 16 bits
40259	+0	KWH	Energy Consumption (same 40257)	Float, upper 16 bits
40260		KWH	Energy Consumption (same 40258)	Float, lower 16 bits
40261	+2	KW	Demand (power)	Float, upper 16 bits
40262		KW	Demand (power)	Float, lower 16 bits

for H8036 products

Modbus Addr	Typical Offset	Units	Description	Data Type
40001	+0	KWH	Energy Consumption, LSW	Integer (mult req'd)
40002	+1	KWH	Energy Consumption, MSW	Integer (mult req'd)
40003	+2	KW	Demand (power)	Integer (mult req'd)
40004	+3	VAR	Reactive Power	Integer (mult req'd)
40005	+4	VA	Apparent Power	Integer (mult req'd)
40006	+5	---	Power Factor	Integer (mult req'd)
40007	+6	VOLTS	Voltage, line to line	Integer (mult req'd)
40008	+7	VOLTS	Voltage, line to neutral	Integer (mult req'd)
40009	+8	AMPS	Current	Integer (mult req'd)
40010	+9	KW	Demand (power), phase A	Integer (mult req'd)
40011	+10	KW	Demand (power), phase B	Integer (mult req'd)
40012	+11	KW	Demand (power), phase C	Integer (mult req'd)
40013	+12	---	Power Factor, phase A	Integer (mult req'd)
40014	+13	---	Power Factor, phase B	Integer (mult req'd)
40015	+14	---	Power Factor, phase C	Integer (mult req'd)
40016	+15	VOLTS	Voltage, phase A-B	Integer (mult req'd)
40017	+16	VOLTS	Voltage, phase B-C	Integer (mult req'd)
40018	+17	VOLTS	Voltage, phase A-C	Integer (mult req'd)
40019	+18	VOLTS	Voltage, phase A-N	Integer (mult req'd)
40020	+19	VOLTS	Voltage, phase B-N	Integer (mult req'd)
40021	+20	VOLTS	Voltage, phase C-N	Integer (mult req'd)
40022	+21	AMPS	Current, phase A	Integer (mult req'd)
40023	+22	AMPS	Current, phase B	Integer (mult req'd)
40024	+23	AMPS	Current, phase C	Integer (mult req'd)
40025	+24	KW	Average Demand	Integer (mult req'd)
40026	+25	KW	Minimum Demand	Integer (mult req'd)
40027	+26	KW	Maximum Demand	Integer (mult req'd)
40257	---	KWH	Energy Consumption	Float, upper 16 bits
40258		KWH	Energy Consumption	Float, lower 16 bits
40259	+0	KWH	Energy Consumption (same 40257)	Float, upper 16 bits
40260		KWH	Energy Consumption (same 40258)	Float, lower 16 bits
40261	+2	KW	Demand (power)	Float, upper 16 bits
40262		KW	Demand (power)	Float, lower 16 bits
40263	+4	VAR	Reactive Power	Float, upper 16 bits
40264		VAR	Reactive Power	Float, lower 16 bits
40265	+6	VA	Apparent Power	Float, upper 16 bits
40266		VA	Apparent Power	Float, lower 16 bits
40267	+8	---	Power Factor	Float, upper 16 bits
40268		---	Power Factor	Float, lower 16 bits
40269	+10	VOLTS	Voltage, line to line	Float, upper 16 bits
40270		VOLTS	Voltage, line to line	Float, lower 16 bits
40271	+12	VOLTS	Voltage, line to neutral	Float, upper 16 bits
40272		VOLTS	Voltage, line to neutral	Float, lower 16 bits
40273	+14	AMPS	Current	Float, upper 16 bits
40274		AMPS	Current	Float, lower 16 bits
40275	+16	KW	Demand (power), phase A	Float, upper 16 bits
40276		KW	Demand (power), phase A	Float, lower 16 bits
40277	+18	KW	Demand (power), phase B	Float, upper 16 bits
40278		KW	Demand (power), phase B	Float, lower 16 bits
40279	+20	KW	Demand (power), phase C	Float, upper 16 bits
40280		KW	Demand (power), phase C	Float, lower 16 bits
40281	+22	---	Power Factor, phase A	Float, upper 16 bits
40282		---	Power Factor, phase A	Float, lower 16 bits
40283	+24	---	Power Factor, phase B	Float, upper 16 bits
40284		---	Power Factor, phase B	Float, lower 16 bits

40285	+26	---	Power Factor, phase C	Float, upper 16 bits
40286		---	Power Factor, phase C	Float, lower 16 bits
40287	+28	VOLTS	Voltage, phase A-B	Float, upper 16 bits
40288		VOLTS	Voltage, phase A-B	Float, lower 16 bits
40289	+30	VOLTS	Voltage, phase B-C	Float, upper 16 bits
40290		VOLTS	Voltage, phase B-C	Float, lower 16 bits
40291	+32	VOLTS	Voltage, phase A-C	Float, upper 16 bits
40292		VOLTS	Voltage, phase A-C	Float, lower 16 bits
40293	+34	VOLTS	Voltage, phase A-N	Float, upper 16 bits
40294		VOLTS	Voltage, phase A-N	Float, lower 16 bits
40295	+36	VOLTS	Voltage, phase B-N	Float, upper 16 bits
40296		VOLTS	Voltage, phase B-N	Float, lower 16 bits
40297	+38	VOLTS	Voltage, phase C-N	Float, upper 16 bits
40298		VOLTS	Voltage, phase C-N	Float, lower 16 bits
40299	+40	AMPS	Current, phase A	Float, upper 16 bits
40300		AMPS	Current, phase A	Float, lower 16 bits
40301	+42	AMPS	Current, phase B	Float, upper 16 bits
40302		AMPS	Current, phase B	Float, lower 16 bits
40303	+44	AMPS	Current, phase C	Float, upper 16 bits
40304		AMPS	Current, phase C	Float, lower 16 bits
40305	+46	KW	Average Demand	Float, upper 16 bits
40306		KW	Average Demand	Float, lower 16 bits
40307	+48	KW	Minimum Demand	Float, upper 16 bits
40308		KW	Minimum Demand	Float, lower 16 bits
40309	+50	KW	Maximum Demand	Float, upper 16 bits
40310		KW	Maximum Demand	Float, lower 16 bits

Note: Modbus addresses in the 4xxxx format follow the Modicon protocol specification for point addressing. The actual address sent is the value shown - 40001. In other words, the leading "4" is omitted, and the remaining 4-digit number is decremented so that point 40001 is requested with a value of zero in the actual modbus communication.

Some modbus implementations require point addresses to be specified beginning at zero or 40000, instead of 40001. The example C code below, for example, requires addresses which correspond to actual values transmitted, so a value of zero is used to request data beginning at modbus address 40001.

In many applications, a single modbus command is used to read all of the data available from the meter. For integers, the beginning address is 40001 (or zero in the actual modbus command), and for floats the first address used would typically be 40259 (or 258 in the actual modbus protocol). Though the first float appears at address 40257, it is not necessary to read this value because it is a duplicate copy of the kwh value (required by the E/node firmware). When a block of data is read, the "typical offset" values index to the data within the block.

USING FLOATING POINT DATA TYPES

The 32-bit floating point values are IEEE-754 standard format. Because Modbus provides only 16-bit registers, two registers must be read to obtain all 32 bits. To eliminate the possibility that data may change between reading the two 16-bit halves of the floating point value, the meter will retain the previously requested 32-bit value. Therefore, the value reported by the combination of two Modbus registers is the value of the data point at the time the first 16-bit portion was requested.

Note: at this time (6-JAN-98) there is no timeout of how long the previously requested float value will be retained in the buffer, so it is important to request a different floating point value before issuing another request for the first one, to avoid re-reading the buffered value. This buffer time-out feature will be added in a future revision of the firmware, so that a single value may be read repetitively.

For systems which use standard IEEE format float point data (such as Intel x86 microprocessors), the 32-bit data can be used directly. In most cases it is not necessary to be concerned with the details of the IEEE-754 format, however,

the IEEE-754 format is a sign bit in the most significant bit, followed by an unsigned 8-bit exponent biased +128, followed by a 23-bit mantissa with an implied one bit, ex 3F800000 -> 1.0 and 43EF1AE1 -> 478.21.

The Energy Consumption data (KWH) appears twice in the floating point data, points 40259 and 40260 are duplicates of 40257 and 40258. In most applications, a Modbus command to read a block of floating point data would specify 40259 as the first address to be read.

USING INTEGER DATA TYPES

Unlike the floating point data type, integer data can only represent whole numbers between 0 to 65535. To convert the number into the actual value it represents, it must be multiplied by a constant. These multipliers are typically small numbers. Many data points require a different multiplier for each product with a different amperage range, but some data types (volts, power factor) require the same multiplier regardless of the product's amperage range.

This table shows the multipliers for all point and amperage ranges:

Addr	Units	100A	300/400A	800A	1600A	2400A
40001	KWH	7.8125e-3	0.03125	0.0625	.125	0.25
40002	KWH	512	2048	4096	8192	16384
40003	KW	0.004	0.016	0.032	0.064	0.128
40004	VAR	0.004	0.016	0.032	0.064	0.128
40005	VA	0.004	0.016	0.032	0.064	0.128
40006	---	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5
40007	VOLTS	0.03125	0.03125	0.03125	0.03125	0.03125
40008	VOLTS	0.015625	0.015625	0.015625	0.015625	0.015625
40009	AMPS	3.9063e-3	0.015625	0.03125	0.0625	0.125
40010	KW	0.001	0.004	0.008	0.016	0.032
40011	KW	0.001	0.004	0.008	0.016	0.032
40012	KW	0.001	0.004	0.008	0.016	0.032
40013	---	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5
40014	---	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5
40015	---	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5	3.0518e-5
40016	VOLTS	0.03125	0.03125	0.03125	0.03125	0.03125
40017	VOLTS	0.03125	0.03125	0.03125	0.03125	0.03125
40018	VOLTS	0.03125	0.03125	0.03125	0.03125	0.03125
40019	VOLTS	0.015625	0.015625	0.015625	0.015625	0.015625
40020	VOLTS	0.015625	0.015625	0.015625	0.015625	0.015625
40021	VOLTS	0.015625	0.015625	0.015625	0.015625	0.015625
40022	AMPS	3.9063e-3	0.015625	0.03125	0.0625	0.125
40023	AMPS	3.9063e-3	0.015625	0.03125	0.0625	0.125
40024	AMPS	3.9063e-3	0.015625	0.03125	0.0625	0.125
40025	KW	0.004	0.016	0.032	0.064	0.128
40026	KW	0.004	0.016	0.032	0.064	0.128
40027	KW	0.004	0.016	0.032	0.064	0.128

It can be helpful in some cases to use "multipliers" as divisors, where the integer value returned by the meter is divided by a number. In many cases, the divisors are a more concise number.

Addr	Units	100A	300/400A	800A	1600A	2400A
40001	KWH	128	32	16	8	4
40002	KWH	1.953e-3	4.8828e-4	2.4414e-4	1.2207e-4	6.1035e-5
40003	KW	250	62.5	31.25	15.625	7.8125
40004	VAR	250	62.5	31.25	15.625	7.8125
40005	VA	250	62.5	31.25	15.625	7.8125
40006	---	32768	32768	32768	32768	32768
40007	VOLTS	32	32	32	32	32
40008	VOLTS	64	64	64	64	64
40009	AMPS	256	64	32	16	8
40010	KW	1000	250	125	62.5	31.25
40011	KW	1000	250	125	62.5	31.25
40012	KW	1000	250	125	62.5	31.25
40013	---	32768	32768	32768	32768	32768
40014	---	32768	32768	32768	32768	32768
40015	---	32768	32768	32768	32768	32768
40016	VOLTS	32	32	32	32	32
40017	VOLTS	32	32	32	32	32
40018	VOLTS	32	32	32	32	32
40019	VOLTS	64	64	64	64	64
40020	VOLTS	64	64	64	64	64
40021	VOLTS	64	64	64	64	64
40022	AMPS	256	64	32	16	8
40023	AMPS	256	64	32	16	8
40024	AMPS	256	64	32	16	8
40025	KW	250	62.5	31.25	15.625	7.8125
40026	KW	250	62.5	31.25	15.625	7.8125
40027	KW	250	62.5	31.25	15.625	7.8125

SUPPORTED MODBUS FUNCTIONS

03 - Read Holding Register

This can be used to read any data within the power meter. Any number of continuous registers may be read, but an exception will be returned if any unimplemented register is specified.

The address field specifies the address beginning at zero for modbus address 40001. See "Modbus Register Addressing" above for details.

Modbus uses a 16-bit CRC, which is the standard CRC16 algorithm with a seed of 65535 (0xFFFF, or all one bits). See the example C code below for two example implementations. A correct CRC field must be transmitted. The E/node meter will not respond to a query with an incorrect CRC, which is the behavior specified by the modbus protocol.

The Modbus master sends an 8-byte request:

```
Byte #1: Address of slave device to read
Byte #2: 03
Byte #3: Address of first data point, MSB
Byte #4: Address of first data point, LSB
Byte #5: Number of points to send, MSB
Byte #6: Number of points to send, LSB
Byte #7: 16-bit CRC, LSB
Byte #8: 16-bit CRC, MSB
```

The E/Node (slave device) will respond with N * 2 + 5 bytes,
where N is the number of points requested:

```
Byte #1: Address of slave device responding
Byte #2: 03
Byte #3: Number of data bytes, N * 2
Byte #4: first data point, MSB
Byte #5: first data point, LSB
Byte #6: second data point, MSB
Byte #7: second data point, LSB
Byte #8: third data point, MSB
Byte #9: third data point, LSB
...
Byte #(N*2+2): last data point, MSB
Byte #(N*2+3): last data point, LSB
Byte #(N*2+4): 16-bit CRC, LSB
Byte #(N*2+5): 16-bit CRC, MSB
```

If any of the data points requested by the 8-byte query from the Master to not exist within the slave device, it will respond with a 5-byte error response. Byte #2 should be checked to determine of the response is a correct output or an error.

```
Byte #1: Address of slave device responding
Byte #2: 83 (131 dec, indicates error condition)
Byte #3: Error Code (02 = invalid address)
Byte #4: 16-bit CRC, LSB
Byte #5: 16-bit CRC, MSB
```

Future Revision: include some example queries and responses from a real meter, with the actual non-zero values and correct CRC's.

06 - Preset Single Register

This function can be used to write to any valid register. Though this is legal for any implemented register, most registers are updated regularly as new readings are made.

Future Revision: add detailed description of command, response, and potential exception responses. See the Modbus reference manual for details.

17 - Report Slave ID

This function will return data which identifies the power meter.

Future Revision: add detailed description of command, response, and potential exception responses. Provide a table of the various responses from each model number

Unused Modbus Functions: These functions are recognized but will always return an exception response. This is due to the fact that all of the data provided by the Veris product is stored in 4xxxx type variables (holding registers). Modbus data types 0xxxx (output coils), 1xxxx (input coils), and 3xxxx (input registers) do not exist within the product, so all attempts to access those types with the following functions will return exception code 02 (Illegal Data Address).

```
01 - Read Coil Status
02 - Read Input Status
04 - Read Input Registers
05 - Force Single Coil
```

Unsupported Modbus Functions. An attempt to use these functions will return exception code 01 "Illegal Function". If the functionality provided by one or more of these Modbus functions is required for your application, please contact Paul Stoffregen via email at paul@veris.com.

```
07 - Read Exception Status
08 - Diagnostics
09 - Program 484
10 - Poll 484
11 - Fetch Comm Event Ctr
12 - Fetch Comm Event Log
13 - Program Controller
14 - Poll Controller
15 - Force Multiple Coils
16 - Preset Multiple Registers
18 - Program 848/M84
19 - Reset Comm Link
20 - Read General Reference
21 - Write General Reference
22 - Mask Write 4X Register
23 - Read/Write 4X Register
24 - Read FIFO Queue
```

WRITING DATA

Any register may be written using the 06 "preset single register" function, but any writes in the second set of registers (float values) will have no effect. The only useful application for writing to a register is to reset (or preset) the kWH accumulator. All other registers will be quickly updated with new data, with overwrites any value written with a Modbus command. To reset the kWH accumulator to zero, the least significant word (LSW) should be written with 0000 first, followed by a write to the most significant word (MSW). This avoids the possibility that the LSW will "roll-over" between the two write functions, which would increment the MSW.

kWH ACCUMULATOR

The kWH accumulator counts the total kilowatt-hour consumption seen by the power meter. This value is supported by non-volatile memory, so that it will not be changed if power is lost temporarily (such as a power outage). Writing to this point (06 function, via registers 40001 and 40002) will alter the permanently stored value.

Applications which require "demand windows" by measuring the kWH consumed within a 15 minute period of time should read the kWH accumulator at the beginning and end of the time period, and subtract the two values. It is not advisable to record the value and the reset the point to zero, because the meter internally records consumption at a very fine scale and may be "just about to increment the accumulator" before it is cleared. Clearing the accumulator at longer intervals, such as once per month, is not likely to cause any significant loss in accuracy.

RTU FRAMING and POTENTIAL ISSUES with SILENT TIME

Modbus Protocol specifies that message framing is accomplished with a 3.5 character silent time. Additionally, Modicon specifies:

"The entire message frame must be transmitted as a continuous stream. If a silent interval of more than 1.5 character times occurs before completion of a frame, the receiving device flushes the incomplete message and assumes that the next byte will be the address field of a new message."

The Veris E/node Modbus meter recognizes the end of a transmission from the Modbus master device by looking for the silent time after the last byte is transmitted. If the master device sends a query with more than approx 2 characters silent time between any of the bytes, it will not be properly recognized and no response will be returned.

RESPONSE TIME

Some systems suspend activity while waiting for a response from a modbus slave device, or have other design parameters which it is important to know how long a slave device will take to answer a query.

The E/node meter will typically respond to a query within 12 ms to 18 ms. All responses will be generated within 40 ms. Because of the Modbus RTU "silent time" requirements for message framing, the slave device can not respond for at least 8 ms. At 9600 baud, each byte takes approx 1 ms to transmit, and the message framing time is 4 ms. The total query time is the sum of the required silent times for message framing, the time to transmit all the bytes at 9600 baud, and the E/node's actual response time.

A worst case example would be a query for all the data available in the full-data unit, using floating point format. First, the master must wait at least 4 ms (many implementations may wait longer) to produce the "silent time" that marks the beginning of the message. 8 ms is required to transmit the 8 byte query. Though it is unlikely that the E/node will take 40 ms to begin transmitting the response, 40 ms is a "worst case" value. A response with 52 modbus points will be 109 bytes long, which will require approximately 109 ms. According to the modbus protocol, the master must wait at least 4 ms (silent time) to recognize the end of the message. (some implementations will not do this and assume that the length count was correct). This gives a worst case time of 165 ms or 0.156 seconds. If the meter responds in the minimum possible time, the total time is reduced to 133 ms, or 0.133 seconds.

However, for queries which request very few points, the baud rate plays a much smaller factor. To request just one point requires 4 ms intial silent time, 8 ms to transmit the bytes, 40 ms (worst case) to respond, 7 ms for the E/node to send the bytes, and a 4 ms silent time to mark the message end, for a total time of 63 ms (worst case). The typical case will be 35 ms to 41 ms. To request 52 points of data could require 2 to 3 seconds by quering one at a time, compared to a worst case of 0.165 seconds using a block request.

The data points within the E/node are low-pass filtered and will require 3-5 seconds to fully respond to a full-scale change that occurs instantly (such as turning on a very large load). However, the variables are internally updated

every 200 ms (5 times per second). Quering the meter more rapidly than 5 times per second will return identical values for some consecutive queries.

ADDITIONAL INFORMATION

See also: example C code (attached)

See also: "Modicon Modbus Protocol Reference Guide", PI-MBUS-300,
Rev J", which is available on the Internet at:

http://www.modicon.com/techpubs/toc7.html