Sample 16-bit Modbus Packet

Sent to Read (16-bit) Process Value

Binary	Hex	Decimal	Purpose
00000001	01	1	Address of Controller
00000011	03	3	Function Read
00000000	00	0	High Byte of Register 100 decimal (Process Temp)
01100100	64	100	Low Byte of Register 100 decimal (Process Temp)
00000000	00	0	High Byte of Number of register to Read (Always 0)
00000001	01	1	Low Byte of Number of registers to Read (Always 1 to 32)
11000101	C5	197	Low byte of CRC
11010101	D5	213	High byte of CRC

The CRC (also a 16 bit wide value) is sent in reverse order, low byte then high byte.

Received from Controller (16-bit) Process Value 74 °F

Binary	Hex	Decimal	Purpose
00000001	01	1	Address of Controller
00000011	03	3	Function Read
00000010	02	2	Number of data bytes returned
00000000	00	0	High Byte Data of register Read
01001010	4A	74	Low Byte Data of register Read
			High Byte Data of register Read (more than one register is requested)
			Low Byte Data of register Read (more than one register is requested)
00111001	39	57	Low byte of CRC
10110011	B3	179	High byte of CRC

Example- To read a 16-bit value in decimal format;

Note: The process value is contained in a 16-bit register. Each register, a 16-bit value, contains a most significant byte, MSB and a least significant byte, LSB. Negative numbers are sent in two's complement format.

In this example, Register 100 MSB = 0 and Register 100 LSB = 74.

16-bit Process Value	
MSB, a 8-bit value	LSB, a 8-bit value
0	74
(MSB x 256) + LSB = Answer	(0 x 256) + 74 = 74

The answer is 74 degrees.

Sample 16-bit Modbus Packet

Sent to Write (16-bit) Set Point of 1,250 °F

Binary	Hex	Decimal	Purpose
00000001	01	1	Controller Address
00000110	06	6	Function Write
00000001	01	1	High Byte of Set Point Register 300 decimal
00101100	2C	44	Low Byte of Set Point Register 300 decimal
00000100	04	4	High Byte of Data to write in Register (New Temp of 80 degrees)
11100010	E2	226	Low Byte of Data to write in Register (New Temp of 80 degrees)
11001011	CB	203	Low byte of CRC
01110110	76	118	High byte of CRC

The CRC (also a 16 bit wide value) is sent in reverse order, low byte then high byte.

Received from Writing to Controller (16-bit) Set Point of 1,250 °F

Binary	Hex	Decimal	Purpose
00000001	01	1	Controller Address
00000110	06	6	Function Write
00000001	01	1	High Byte of Set Point Register 300 decimal
00101100	2C	44	Low Byte of Set Point Register 300 decimal
00000100	04	4	High Byte of Data to write in Register (New Temp of 134 degrees)
11100010	E2	226	Low Byte of Data to write in Register (New Temp of 134 degrees)
11001011	CB	203	Low byte of CRC
01110110	76	118	High byte of CRC

Example- To write a 16-bit value in decimal format;

The set point value of the Series F4 is contained in register 300. To determine the most significant byte (MSB), divide the set point (SP) by 256. To determine the least significant byte (LSB), subtract from the SP the integer results of multiplying the MSB by 256.

SP = 1,250
Register 300 is written with a value of 1,250

Set Point Value is a 16-bit value	
Integer of 1250/256 = 4	Remainder is 1250 - (4 x 256) = 226
MSB, a 8-bit value	LSB, a 8-bit value
4	226
(MSB x 256) + LSB = Answer	(4 x 256) + 226 = 1250

If the controller displays with a decimal point, then the Modbus value has an assumed decimal point. As an example, a set point of 125.0 is sent as 1250 when the decimal register 606 is set to a value of 1. When register 606 has a value of 0, then a value of 1250 means one thousand two hundred and fifty degrees.

1241 Bundy Blvd
Winona, MN 55987
Telephone (507) 494-5656

Cyclical Redundancy Checksum (CRC) Algorithm

This C routine, calc_crc(), calculates the cyclical redundancy checksum, CRC, for a string of characters. The CRC is the result of dividing the string by 0xA001. Modbus applications calculate the packet's CRC then append it to the packet.

```
#define POLYNOMIAL 0xA001;
unsigned int calc_crc(unsigned char
*start_of_packet, unsigned char
*end_of_packet)
{
unsigned int crc;
unsigned char bit_count;
unsigned char *char_ptr;
/* Start at the beginning of the packet */
char_ptr = start_of_packet;
/* Initialize CRC */
crc = 0xffff;
/* Loop through the entire packet */
do{
/* Exclusive-OR the byte with the CRC */
crc ^= (unsigned int)*char_ptr;
/* Loop through all 8 data bits */
bit_count = 0;
do{
/* If the LSB is 1, shift the CRC and XOR
the polynomial mask with the CRC */
if(crc & 0x0001){
crc >>= 1;
crc ^= POLYNOMIAL;
}
/* If the LSB is 0, shift the CRC only */
else{
crc >>= 1;
}
} while(bit_count++ < 7);
} while(char_ptr++ < end_of_packet);
return(crc);
}
```