

# Love Controls Implementation of the Modbus RTU protocol

If you have a software application program that uses Modbus, you can skip to the Modbus Register listing in this document for the address information your program will need. The rest of this section provides the information you will need to write your own software application using Modbus RTU.

## Modbus RTU

Messages are coded in eight-bit bytes, with one start bit and one stop bit, no parity bit is used (8,N,1). Negative values are written in twos compliment format. Parameters are read from, or written to, 16 bit registers (two-byte) using an **absolute** address.

Message packets are separated by a pause of at least 3 character times(30 bits). The packet must be sent in a continuous stream with an inter-character interval of no greater than 1.5 character times (15 bits). Messages that begin without a 30 bit silent interval or have an inter-character interval greater than 15 bits are ignored.

Each message packet uses the following syntax.

The first byte is a 1 byte instrument address, Love Controls uses addresses 0x01 through 0xFF. The second byte is the command, read status (0x01 or 0x02), read register (0x03 or 0x04), write register (0x06 or 0x10), or loop back (0x08).

The next n bytes are the register address and / or data.

The last two bytes are for error checking and contain the CRC16 generated on the message bytes.

Address	Command	Reg. / Data	CRC
##	##	####.....	####

## Read Multiple Registers Command (0x03 or 0x04)

This command actually reads a single register, the number of registers must be fixed at (0x0001).

Message sent to instrument:

##	03	##	##	00	01	##	##
Instrument Address	Read Command	High byte Starting register address	Low byte	High byte Number of registers	Low byte	Low byte CRC	High byte CRC

Message returned by instrument:

##	03	02	##	##	##	##
Instrument Address	Read Command	Number of bytes	High byte Register	Low byte Data	Low byte CRC	High byte CRC

**Example:** Read the Process Value register (0x0001) from instrument at address 50 (0x32).

Sent: 32 03 00 01 00 01 D0 09

Received: 32 03 02 00 64 BD AB

Data: 100 (0x0064)

\

## Write to a Single Register Command (0x06)

This command writes to a single register, the instrument will echo back the message if the data was accepted.

Message sent to instrument:

##	06	##	##	##	##	##	##
Instrument Address	Write Command	High byte Register Address	Low byte Register Address	High byte Register Data	Low byte Register Data	Low byte CRC	High byte CRC

**Example:** Write 257 (0xC8) to the SP1 register (0x0101) of instrument at address 50 (0x32).

Sent: 32 06 01 01 00 C8 DD A3

Received: 32 06 01 01 00 C8 DD A3

## Write Multiple Registers Command (0x10)

This command actually writes to a single register, the number of registers must be fixed at (0x0001) and the number of data bytes fixed at (0x02).

Message sent to instrument:

##	10	##	##	00	01	02	##	##	##	##
Instrument Address	Write Cmd.	H. byte Starting	L. byte Register	H. byte Number of	L. byte Regs.	Data Bytes	H. byte Data	L. byte Data	L. byte CRC	H. byte CRC

Message returned by instrument:

##	10	##	##	00	01	##	##
Instrument Address	Write Command	High byte Starting register	Low byte	High byte Number of registers	Low byte	Low byte CRC	High byte CRC

**Example:** Write 175 (0xAF) to the SP1 register (0x0101) of instrument at address 50 (0x32).

Sent: 32 10 01 01 00 01 02 00 AF B7 CC

Received: 32 10 01 01 00 01 54 36

## Read Coil or Input Status Command (0x01 or 0x02)

This command actually reads a single bit, the number of points must be fixed at (0x0001).

Message sent to instrument:

##	02	##	##	00	01	##	##
Instrument Address	Read Command	High byte Starting	Low byte address	High byte Number of points	Low byte	Low byte CRC	High byte CRC

Message returned by instrument:

##	02	01	##	##	##
Instrument Address	Read Command	Number of bytes	Single Byte Register Data	Low byte CRC	High byte CRC

The single byte of data returned contains the requested Status bit in the LSB (0000000#), status is indicated as 1 = ON and 0 = OFF.

**Example:** Read the AL1 state (0x0003) from instrument at address 50 (0x32).

Sent: 32 02 00 03 00 01 4C 09

Received: 32 02 01 01 6F 0C

Data: 1 (0x01)

## Loop Back Command (0x08)

This command causes the instrument to echo back the message it receives. It is useful as a diagnostic tool to check wiring, baud rate, instrument address etc.

Message sent to the instrument:

##	08	##	##	##	##
Instrument Address	Loop back Command	High byte Data	Low byte Data	Low byte CRC	High byte CRC

**Example:** Perform a loop back test on the instrument at address 50 (0x32)

Sent: 32 08 12 34 82 29

Received: 32 08 12 34 82 29

## Exception Codes

Commands that cannot be processed by an instrument will cause an exception code to be returned. These messages are identified by the command byte returned having its high bit set, i.e. +(0x80).

Exception codes supported by Love Controls instruments are as follows:

0x01 Illegal Command

0x02 Illegal Address

0x03 Illegal Data

Message returned by instrument:

##	##	##	##	##
Instrument Address	Command + <b>0x80</b>	Exception Code	Low byte CRC	High byte CRC

Messages sent with the wrong CRC, format or timing are ignored by the instrument. The broadcast command (write to instrument address 0x00) is not supported.

**Example:** Command 02 is not supported (Exception 01)

Sent: 32 02 01 07 00 01 0C 34

Received: 32 **82 01** 71 6F

**Example:** Register 258 (0x0102) is inactive (Exception 02)

Sent: 32 04 01 02 00 01 94 35

Received: 32 **84 02** 32 CE

**Example:** Cannot write 10,000 (0x2710) to SP1 (0x0101), Illegal data value (Exception 03)

Sent: 32 06 01 01 27 10 C6 09

Received: 32 **86 03** F2 6E

## CRC Calculation

The last two bytes of all messages contain the CRC value. The CRC is calculated for the message being transmitted and is appended to the message. The receiving device recalculates a CRC for the message it receives. If the two CRC values do not match, the message is ignored by the receiver.

The CRC is calculated by first loading a 16 bit register with all 1's (0xFFFF), this register will hold the final CRC value. A second 16 bit register is loaded with a constant (0xA001). Each eight bit character is then applied to the CRC register using the following procedure.

1. The eight bit character is Exclusive ORed with the CRC register.
2. The CRC register is then shifted one bit toward the LSB (least significant bit) and a zero placed in the MSB (most significant bit).
3. The LSB from the result of the Exclusive OR between the CRC and the eight bit character is then examined, if the LSB is a 1 then the CRC is Exclusive ORed with the constant, if the LSB is a 0 no Exclusive OR is performed.
4. The eight bit character is then shifted one bit toward of the LSB and a zero is placed in the MSB.

Steps 1 through 4 are repeated for all eight bits of the character, then the process is repeated for the next character. Once all the message characters have been processed the CRC register will contain the final CRC value. When the CRC is appended to the message, the byte order is reversed. As an example, a CRC of **0x0A84** will be appended as **0x840A**

A sample CRC calculation program written for QBASIC is supplied on the next page.

```
REM          CRC16.BAS
```

```
REM if message for crc is 010300000001  crc = 0A84 (0A high byte, 84 low byte)  
REM if message for crc is 320301010001  crc = D1F5
```

```
crc = &HFFFF&  
polynomial = &HA001&  
msg$ = "320301010001"  
ptr = 1
```

```
DO WHILE ptr < LEN(msg$)  
num$ = "&h" + (MID$(msg$, ptr, 2))  
num = VAL(num$)  
ptr = ptr + 2  
cnt = 8
```

```
    DO WHILE cnt > 0  
    temp = num XOR crc  
    crc = INT(crc / 2)  
    IF (temp AND 1) THEN crc = crc XOR polynomial  
    num = INT(num / 2)  
    cnt = cnt - 1  
    LOOP
```

```
LOOP  
PRINT "crc", HEX$(crc): REM The CRC is generated in high byte, low byte order  
END:          REM but the order is reversed in the actual message.
```

