C130-FL-PMAG

Modbus Protocoll

Technical Manual

Cod. C130-FL-Modbus_UK_M1

English

Prod. Rev: 1.0 - Manual Rev: 1.0

CONTENTS

1 THE SERIAL TRANSMISSION MODES

There are two MODBUS serial transmission modes, ASCII and RTU. In RTU mode use of 8bit binary characters, ASCII mode, use of 7bit ASC characters. The RTU mode, the high byte of a 4 bit and low-4 bit separate into two bytes, it can be change to transmit byte ASCII mode. For example RTU mode, data 0x1A, then the ASCII mode is 0x31 0x41 2 bytes, so the frame length of ASCII mode is double of the RTU mode .

RTU transmission mode of the data frame is CRC checksum, ASCII mode with LRC checksum.

The following table summarizes the difference between two transmission modes:

TRANSMISSION MODE	**ASCII (7 bit)**	**RTU (8 bit)**
Code format	ASCII code (‘0’-‘9’ ‘A’ -‘F’)	8bit binary characters (0x00 – 0xff)
Start bit Data bits Parity bit stop bit	1 7、8 NONE/even/odd 1、2	1 8 NONE/even/odd 1、2
Error Check Field	LRC	CRC16

2 Register and message format

List some of Register and message format

Register type	**message length**	**Register qty**	**description**
COIL bit	1	-	COIL Variable(ON OFF)
FLOAT	32 bit	2	32bit float point number(IEEE754format)
INT	16 bit	1	unsigned INT(0x0 – 0xFFFF)
LONG	32 bit	2	unsigned long INT(0x0 – 0xFFFFFFFF)

2.1 COIL

COIL Variable 0xFF00 -> ON 0x0000 -> OFF

2.2 FLOAT

Apply 2 Register store single -precision IEEE754 format float point number。

Every float point number include 4 BYTE Specifically defined as follows

SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM

S: signed bit 0->positive 1->negitive

E: exponent

M:The fractional part of mantissa

For example 0xC1480000 = -12.5

2.3 INT

Apply1 Register to store a unsigned INT number。

For example 0x0025 = 37 0x1234 = 4660

2.4 LONG

Apply2 Registers to store a unsigned long INT number.

For example 0x12345678 = 305419896

3 Message format definition

3.1 CMD=0x03(read 1 or more Registers)

For example the message is to read instant flow Message slave address=1。

Note the instant flow Register Start address=0x0595 however the Message Register of Start address should be 0x0595-0x0001 = 0x0594

Query Master->slave

Message format name	**RTU example data(HEX)**	**ASC example data(HEX)**
Head of package	NONE	3A
Slave address	01	30 31
Function code	03	30 33
Register Start address high BYTE	02	30 32
Register Start address low BYTE	52	35 32
Register qty high BYTE	00	30 30
Register qty low BYTE	02	30 32
ErrorCheckField	64 62	41 36
End of package	NONE	0D 0A

Response Slave->Master

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	03	30 33
message length	04	30 34
Register0x0253 data high BYTE	C1	43 31
Register0x0253 data low BYTE	48	34 38
Register0x0254 data high BYTE	00	30 30
Register0x0254 data low BYTE	00	30 30
Error Check Field	47 D9	45 46
End of package	NONE	0D 0A

The Response will return IEEE754 format instant flow value of C1 48 00 00 = -12.5

3.2 CMD=0x05(write COILVariable)

the case of data to remove the accumulated flow Message slave address=1。

Note Clear total Register Start address=0x0003 however the Message of Register Start address should be 0x0003-0x0001 = 0x0002

Query Master->slave

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	05	30 35
Register Start address high BYTE	00	30 30
Register Start address low BYTE	02	30 32
COIL Variable high BYTE	FF	46 46
COIL Variable low BYTE	00	30 30
Error Check Field	2D FA	46 39
end of package	NONE	0D 0A

Response Slave->Master

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	05	30 35
RegisterStart addresshighBYTE	00	30 30
RegisterStart addresslowBYTE	02	30 32
COILVariablehighBYTE	FF	46 46
COILVariablelowBYTE	00	30 30
Error Check Field	2D FA	46 39
end of package	NONE	0D 0A

3.3 CMD=0x06(write a single Register)

the case of data to write flow unit=m3/h Message slave address=1。

NOTE flow unit Register Start address=0x0042 However the Message Register Start address should be 0x0042-0x0001 = 0x0041

Query Master->Slave

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	06	30 36
Register Start address high BYTE	00	30 30
Register Start address low BYTE	41	34 31
Variable high BYTE	00	30 30
Variable low BYTE	13	31 33
Error Check Field	98 13	41 35
end of package	NONE	0D 0A

Response Slave->Master

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	06	30 36
RegisterStart address high BYTE	00	30 30
Register Start address low BYTE	41	34 31
Variable high BYTE	00	30 30
Variable low BYTE	13	31 33
Error Check Field	98 13	41 35
end of package	NONE	0D 0A

3.3 CMD=0x10(write many Registers)

the case of data to write damping time=3s Message slave address=1。

Note The damping time Register Start address=0x0189 however the Message Register Start address should be 0x0189-0x0001 = 0x0188

Query Master->slave

message format name	**RTUexample data(HEX)**	**ASCexample data(HEX)**
Head of package	NONE	3A
slave address	01	30 31
function code	10	31 30
RegisterStart address high BYTE	01	30 31
RegisterStart address low BYTE	88	38 38
Register qty high BYTE	00	30 30
Register qty low BYTE	02	30 32
message length	04	30 34
To write Register0x0189 into high BYTE	40	34 30
To write Register0x0189 into low BYTE	40	34 30
To write Register0x018A into high BYTE	00	30 30
To write Register0x018A into low BYTE	00	30 30
Error Check Field	E3 ED	45 38
end of package	NONE	0D 0A

The case of 4 data BYTE is IEEE754 format float point number40 40 00 00 = 3.0

Response slave ->Master

message format name	RTUexample data(HEX)	ASCexample data(HEX)
Head of package	NONE	3A
slave address	01	30 31
function code	10	31 30
RegisterStart address high BYTE	01	30 31
RegisterStart address low BYTE	88	38 38
Register qty high BYTE	00	30 30
Register qty low BYTE	02	30 32
Error Check Field	C0 1E	36 43
end of package	NONE	0D 0A

3.4 Exception Response

If an error is detected in the content of the query (excluding parity errors and Error Check mismatch), the function code will be modified to indicate that the response is an error response (called an exception response), and the data bytes will contain a code that describes the error.

For example, if flow unit to be setup as Hz by the reason the flow meter unable to use Hz as flow unit so return Exception Response。

Exception Response Slave->Master

message format name	RTUexample data(HEX)	ASCexample data(HEX)
Head of package	NONE	3A
slave address	01	30 31
function code	86	38 36
Error code	43	34 33
ErrorCheckField	03 91	39 31
end of package	NONE	0D 0A

Note 1 Exception Response function code=Query function code+0x80

2 Detail Error code to reference **Appendix 1: Constant table: Error code**

4 Data Error Check Field algorithm

4.1 LRC CHECK

// LRC CHECK Range From “slave address”to the last byte before LRC Error Check Field.

```
void LRC(unsigned char *buf, unsigned int len)
{
    unsigned int i;
    LRC = 0;
    for (i=0; i<len; i++)
    {
 LRC += buf[i];
 }
    LRC = 0xff - LRC;
    LRC++;
}
```

4.2 CRC16 CHECK

```
const unsigned char  TAB_CRC_H[] = {
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40,0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,
    0x00,0xC1,0x81,0x40,0x01,0xC0,0x80,0x41,0x01,0xC0,0x80,0x41,0x00,0xC1,0x81,0x40
};


const unsigned char  TAB_CRC_L[] = {
    0x00,0xC0,0xC1,0x01,0xC3,0x03,0x02,0xC2,0xC6,0x06,0x07,0xC7,0x05,0xC5,0xC4,0x04,
    0xCC,0x0C,0x0D,0xCD,0x0F,0xCF,0xCE,0x0E,0x0A,0xCA,0xCB,0x0B,0xC9,0x09,0x08,0xC8,
    0xD8,0x18,0x19,0xD9,0x1B,0xDB,0xDA,0x1A,0x1E,0xDE,0xDF,0x1F,0xDD,0x1D,0x1C,0xDC,
    0x14,0xD4,0xD5,0x15,0xD7,0x17,0x16,0xD6,0xD2,0x12,0x13,0xD3,0x11,0xD1,0xD0,0x10,
    0xF0,0x30,0x31,0xF1,0x33,0xF3,0xF2,0x32,0x36,0xF6,0xF7,0x37,0xF5,0x35,0x34,0xF4,
    0x3C,0xFC,0xFD,0x3D,0xFF,0x3F,0x3E,0xFE,0xFA,0x3A,0x3B,0xFB,0x39,0xF9,0xF8,0x38,
    0x28,0xE8,0xE9,0x29,0xEB,0x2B,0x2A,0xEA,0xEE,0x2E,0x2F,0xEF,0x2D,0xED,0xEC,0x2C,
    0xE4,0x24,0x25,0xE5,0x27,0xE7,0xE6,0x26,0x22,0xE2,0xE3,0x23,0xE1,0x21,0x20,0xE0,
```

```
    0xA0,0x60,0x61,0xA1,0x63,0xA3,0xA2,0x62,0x66,0xA6,0xA7,0x67,0xA5,0x65,0x64,0xA4,
    0x6C,0xAC,0xAD,0x6D,0xAF,0x6F,0x6E,0xAE,0xAA,0x6A,0x6B,0xAB,0x69,0xA9,0xA8,0x68,
    0x78,0xB8,0xB9,0x79,0xBB,0x7B,0x7A,0xBA,0xBE,0x7E,0x7F,0xBF,0x7D,0xBD,0xBC,0x7C,
    0xB4,0x74,0x75,0xB5,0x77,0xB7,0xB6,0x76,0x72,0xB2,0xB3,0x73,0xB1,0x71,0x70,0xB0,
    0x50,0x90,0x91,0x51,0x93,0x53,0x52,0x92,0x96,0x56,0x57,0x97,0x55,0x95,0x94,0x54,
    0x9C,0x5C,0x5D,0x9D,0x5F,0x9F,0x9E,0x5E,0x5A,0x9A,0x9B,0x5B,0x99,0x59,0x58,0x98,
    0x88,0x48,0x49,0x89,0x4B,0x8B,0x8A,0x4A,0x4E,0x8E,0x8F,0x4F,0x8D,0x4D,0x4C,0x8C,
    0x44,0x84,0x85,0x45,0x87,0x47,0x46,0x86,0x82,0x42,0x43,0x83,0x41,0x81,0x80,0x40
};
```

// CRC CHECK Range ： From“slave address”to the last byte before CRC Error Check Field.

```
void CRC(unsigned char *buf, unsigned int len)
{
    unsigned int i;
    unsigned char CRC_H , CRC_L , index, ch;
    CRC_H = 0xff;
    CRC_L = 0xff;
    for (i=0; i<len; i++)
    {
        ch = buf[i];
        index = CRC_H ^ ch;
        CRC_H = CRC_L ^ TAB_CRC_H[index];
        CRC_L = TAB_CRC_L[index];
    }
}
```

5 Flow meter Variable(slave address)definition

Following is a list of instrument Variable information the data are HEX type

Variable name	Register address	Register length	Read instruction	Write instruction
COILtype				
Clear Total	0003	---	---	05
INT type				
Flowunit (Appendix2:Constant table:flow unit)	0042	0001	03	06
Total unit (Appendix 2: Constant table:flow unit)	0046	0001	03	06
LONG type				
Expansion of the positive accumulated	0309	0002	03	---
positive accumulated	0311	0002	03	---
Expansion of the negtive accumulated	0313	0002	03	---
Negative accumulated	0315	0002	03	---
FLOAT type				
Main Variable(instant flow)	0253	0002	03	---
damping times	0189	0002	03	10
Cut off %	0197	0002	03	10
Qmax(m3/L)	0209	0002	03	10
4-20mAcurrent test(mA)	0143	0002	---	10
Output current mA	0203	0002	03	---
Output Frequency Hz	0229	0002	03	---

Note The flow total calculation is as follows:

Suppose read out the "expansion of positive cumulative” = 2 “positive cumulative” = 1234

The total positive flow = 2 * 1000,0000 + 1234 = 20001234

6 Appendix 1: Constant table： Error code

0x01: Invalid instruction code

0x02: Invalid Register address

0x30: parameter upper limit

0x31: parameter of super threshold

0x32: parameter option item error

0x40: Invalid Register length

0x41: Register unable to support current instruction code

0x42: Register unassigned

0x43: flow unit absent

0x44: Total unit absent

0x45: the highest Frequency output upper limit

0x46: the lowest Frequency output of super threshold

0x47: the high flow speed upper limit

0x48: duty cycle upper limit

7 Appendix 2: Constant table： flow unit

inH2O	1
inHg	2
ftH2O	3
mmH2O	4
mmHg	5
psi	6
bar	7
mbar	8
g/Sqcm	9
Kg/Sqcm	10
Pa	11
kPa	12
torr	13
atm	14
Cuft/min	15
gal/min	16
L/min	17
Impgal/min	18
Cum/h	19
ft/s	20
m/s	21
gal/s	22
MMgal/d	23
L/s	24
ML/d	25
Cuft/s	26
Cuft/d	27
Cum/s	28

Cum/d	29
Impgal/h	30
Impgal/d	31
degC	32
degF	33
degR	34
Kelvin	35
mV	36
ohm	37
Hz	38
mA	39
gal	40
L	41
Impgal	42
Cum	43
ft	44
m	45
bbl	46
in	47
cm	48
mm	49
min	50
s	51
h	52
d	53
cSt	54
cP	55
uMho	56
%	57
V	58

pH	59
g	60
kg	61
MetTon	62
lb	63
STon	64
LTon	65
g/s	70
g/min	71
g/h	72
Kg/s	73
Kg/min	74
Kg/h	75
Kg/d	76
MetTon/min	77
MetTon/h	78
MetTon/d	79
lb/s	80
lb/min	81
lb/h	82
lb/d	83
STon/min	84
STon/h	85
STon/d	86
LTon/h	87
LTon/d	88
SGU	90
g/Cucm	91
Kg/Cum	92
lb/gal	93

lb/Cuft	94
g/mL	95
kg/L	96
g/L	97
lb/Cuin	98
STon/Cuyd	99
degTwad	100
degBrix	101
degBaum hv	102
degBaum lt	103
degAPI	104
% sol-wt	105
% sol-vol	106
degBall	107
proof/vol	108
proof/mass	109
bush	110
Cuyd	111
Cuft	112
Cuin	113
m/h	120
Cuft/h	130
Cum/min	131
bbl/s	132
bbl/min	133
bbl/h	134
bbl/d	135
gal/h	136
Impgal/s	137
L/h	138

% Stm Qual	150
ft.in16	151
Cuft/lb	152
pF	153
% plato	160
KW	161
MW	162
KWh	163
MWh	164
gal/d	235
hL	236
Mpa	237
inH2O @4DegC	238
mmH2O @4DegC	239
MetTon/s	240
ML/s	241
ML/min	242
ML/h	243
L/d	244
g/d	245
ML	246
KJ	247
MJ	248
GJ	249
KJ/h	250
MJ/h	251
GJ/h	252

Company With Quality System Certified
UNI EN ISO 9001:2008

CEAM Control Equipment srl

Headquarters:
Via Val D'Orme No. 291
50053 Empoli (Firenze) Italy
Tel. (+39) 0571 924082 - Fax. (+39) 0571 924505
Skype Name: ceam_info

Internet:
Corporate: www.ceamgroup.com
Technical Support: www.ceamsupport.it

E.mail:

General: info@ceamgroup.it
Sales Support: sales@ceamgroup.it

Local Reseller: