T9C series counter communication protocol

T9C series meters use Modbus RTU communication protocol to carry out RS485 half-duplex communication, read function number 03H, write function number 10H, adopt 16-bit CRC check, the meter will not return check error.

Data frame format:

Start bit	Data bit	Stop bit	Check bit
1	8	1	NO

Communication exception handling:

When responding abnormally, set the highest bit of the function number to 1. For example, if the function number requested by the master is 04H, the corresponding item of the function number returned from the slave is 84H.

Error type code:

01H --- Illegal function code: The instrument does not support the received function number.

02H --- Illegal data location: The data location specified by the host exceeds the range of the instrument.

03H --- illegal data value: the data value sent by the host exceeds the data range corresponding to the instrument.

1. Read multiple registers

Example: When the host reads the count value 40015 (0EH) ~ 40016 (0FH): (for example: the count value is 123456)

The address codes of the count value are 0EH and 0FH, because the count value is a 32-bit floating point number (4 bytes) and occupies 2 data registers. The IEEE-574 standard hexadecimal memory code for decimal floating point number 123456 is 47F12000H, and the transmission format is 0020F147H.

Host request (read multiple registers)							
1	2	3	4	5	6	7	8
Meter address	Function number	Start address		Number of data		CRC16 code	
		High bit	Low bit	High bit	Low bit	Low bit	High bit
01H	03H	00H	0EH	00H	02H	A5H	C8H

Slave responds normally (read multiple registers)								
1	2	3	4	5	6	7	8	9
Meter address	Function number	Number of data bytes	Data 1		Data 2		CRC16 code	
			High bit	Low bit	High bit	Low bit	Low bit	High bit
01H	03H	04H	00H	20H	F1H	47H	FEH	5BH

Abnormal response to the function number: (for example, the host request function number is 04H)

Slave abnormal response (read multiple registers)				
1	2	3	4	5
Meter address	Function number	Error code	CRC16 code Low bit	CRC16 code High bit
01H	84H	01H	82H	C0H

二、 Write multiple registers

Example: When the host writes the PSCL value from 40009 (08H) to 40010 (09H): (for example: effective value 1.000)

The address codes of PSCL are 08H and 09H, because PSCL is a 32-bit floating point number (4 bytes) and occupies 2 data registers. The IEEE-574 standard hexadecimal memory code for decimal floating point number 1.000 is 3F800000H, and the transmission format is 0000803FH.

Host request (read multiple registers)												
1	2	3	4	5	6	7	8	9	10	11	12	13
Meter address	Function number	Start address		Number of registers		Number data bytes	Data 1		Data 2		CRC16 code	
		High bit	Low bit	High bit	Low bit		High bit	Low bit	High bit	Low bit	Low bit	High bit
01H	10H	00H	08H	00H	02H	04H	00H	00H	80H	3FH	D3H	D9H

Slave responds normally (write multiple registers)							
1	2	3	4	5	6	7	8
Meter address	Function number	Start address		Number of registers		CRC16 code	
		High bit	Low bit	High bit	Low bit	Low bit	High bit
01H	10H	00H	08H	00H	02H	C0H	0AH

Data location error response (for example: host request write address code is 50H)

Slave responds normally (write multiple registers)				
1	2	3	4	5
Meter address	Function number	Error code	CRC16 code Low bit	CRC16 code High bit
01H	90H	02H	CDH	C1H

T9C related parameter address coding table

Address code	Function number	Parameter name	Ranges	Read and write allowed	Remarks
40001(00H)*	03/16	2-step set value SET2	0~999999	R/W	Unit same as measured value
40002(01H)*					
40003(02H)**	03/16	1 set value SET1	0~999999	R/W	Unit same as measured value
40004(03H)**					
40005(04H)*	03/16	OUT2 output reset time	0.01~99.99	R/W	
40006(05H)*					
40007(06H)**	03/16	OUT1 output reset time	0.00~99.99	R/W	
40008(07H)**					
40009(08H)	03/16	Measure PSCL	0.0001~99.9999	R/W	
40010(09H)					
40011(0AH)	03/16	Transmitting output upper limit HLOP	-99999~999999	R/W	Unit same as measured value
40012(0BH)					

40013(0CH) 40014(0DH)	03/16	Transmission output lower limit LLOP	-99999~999999	R/W	Unit same as measured value
40015(0EH) 40016(0FH)	03	Count value	-99999~999999	R	
	Keep				
40021(14H)	03/16	Input mode IN	0: U； 1: d； 2: Ud-A 3: Ud-B； 4: Ud-C	R/W	
40022(15H)*	03/16	Output mode OUTM	0: n； 1: F； 2: C； 3: r； 4: L； 5: K； 6: d； 7: H	R/W	
40023(16H)	03/16	Counting speed CPS	0: 5； 1: 30； 2: 1K； 3: 5K	R/W	
40024(17H)	03/16	Decimal point DP	0:------； 1:-----.-； 2:----.--； 3: ---.---； 4: --.----；	R/W	
40025(18H)	03/16	Transmitting output type LOPT	0: 0-20； 1: 4-20	R/W	
40026(19H)	03/16	NPN / PNP input SIG	0: nPn； 1: PnP	R/W	
40027(1AH)	03/16	Power failure memory selection DATA	0: rEC； 1: CLEr	R/W	
40028(1BH)	03/16	Key protection level KP	0:0； 1:1； 2:2； 3:3； 4:4；	R/W	
40029(1CH)	03/16	Communication reset meter	0~1	R/W	**(Write 1 reset)**
40030(1DH)	03	Baud rate BPS	0: 2400； 1: 4800 2: 9600； 3: 19200	R	
40031(1EH)	03	Meter address AdrS	0~255	R	
40032(1FH)	03	Measurement status indication	0~31	R	Note①
40033(20H)	03	Instrument model character	C0H~C2H	R	Note②
	Keep				

R: Read only; R / W: Read and write.

Note: The parameters with an asterisk are only available for T9C-1P / T9C-2P. If you read and write to T9C-N, it is regarded as invalid parameter address. The parameters with ** are T9C-2P instruments.

Note①: Measurement status indication

Bit15~Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
-	-	-	RESET	KP	Overflow indication （FFFFFF/------）	OUT2 /OUT	OUT1
0	0	0	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1

Note②: Characteristic words of the instrument model

Instrument model	T9C-N	T9C-1P	T9C-2P
Communication value	C0H	C1H	C2H

Program for converting floating point numbers represented by 4 byte character inner codes

into decimal floating point numbers

```
float    BytesToFloat(unsigned char *pch)
{
      float result;
      unsigned char *p;
      p=(unsigned char *)&result;
      *p=*pch; *(p+1)=*(pch+1); *(p+2)=*(pch+2); *(p+3)=*(pch+3);
      return result;
}
```

Decimal floating point number is converted into a 4-byte character inner code representation program according to the IEEE-754 standard

```
void FloatToChar(float Fvalue, unsigned char *pch)
{
   unsigned char *p;
   p=(unsigned char*)&Fvalue;
   *pch=*p;    *(pch+1)=*(p+1);    *(pch+2)=*(p+2);    *(pch+3)=*(p+3);
}
```

16-bit CRC check code acquisition program

```
unsigned int Get_CRC(unsigned char    *pBuf, unsigned char    num)
{
      unsigned char i,j;
      unsigned int wCrc = 0xFFFF;
      for(i=0; i<num; i++)
      {
            wCrc ^= (unsigned int)(pBuf[i]);
            for(j=0; j<8; j++)
            {
                  if(wCrc & 1){wCrc >>= 1; wCrc ^= 0xA001; }
                  else
                        wCrc >>= 1;
            }
      }
          return wCrc;
}
```