MODBUS-RTU PROTOCOL SPECIFICATION FOR DSZ15DZMOD V1.6

1 transmission characteristic
1.1 bit transmission
1.1.1 LSB transmission order: LSB
1.1.2 characteristic bit:8 data bits,1 stop bit, no check bit
1.1.3 baud rate: 9600

1.2 byte transmission
1.2.1 Byte transfer order of a single register: data or address transmitted from Hi to Lo (2 byte).
1.2.2 Transfer order of multiple registers: (n*2byte, n>1) from Lo register address to Hi register address.
1.2.3 CRC check order: from Lo CRC check byte to Hi CRC check byte (2byte)

2 order format
2.1 read variate data (function code 0×04)
2.1.1 master

Address field	Function code 04H	start address Hi	start address Lo	number of register Hi	number of register Lo	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

Example: send mater of '00 04 00 48 00 04 70 0E' or 'CC 04 00 48 00 04 61 C2' to read 'total import active energy' or 'total export active energy' (meter address is 0xCC for all the examples).

2.1.2 slave (correct)

Address field	Function code 04H	number of bytes	register 1 Hi	register 1 Lo	register 2 Hi	register 2 Lo	.....	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte		1 byte	1 byte

Example: the meter will answer 4.61 kWh for total import active energy and 3.68kWh for total export active energy. The correct slave is 'CC 04 08 00 00 01 CD 00 00 01 70 CF D7'.

2.1.3 slave (incorrect)

Address field	Function code 86H	Error code	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte

Example: if send incorrect function code 0x05 'CC 05 00 48 00 04 5C 02', it will answer 'CC 86 01 12 5F'

2.2 read parameter data (function code 0×03
2.1.1 master

Address field	Function code 03H	start address Hi	start address Lo	number of register Hi	number of register Lo	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

Example: send mater of '00 03 00 56 00 02 25 CA' or 'CC 03 00 56 00 02 34 06' to read pulse mode.

2.2.2 slave (correct)

Address field	Function code 03H	number of bytes	register 1 Hi	register 1 Lo	register 2 Hi	register 2 Lo	.....	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte		1 byte	1 byte

Example: if pulse mode is 0x0x, the correct slave should be 'CC 03 04 00 00 00 02 67 3E'.

2.2.3 slave (incorrect)

Address field	Function code 86H	Error code	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte

2.3 read parameter data (function code 0×10)

2.3.1 master

Address field	Function code 10H	start address Hi	start address Lo	number of register Hi	number of register Lo	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte

Example: send mater of '00 10 00 14 00 02 04 00 00 00 2A 76 73' or 'CC 10 00 14 00 02 04 00 00 00 2A B5 20' to change meter address to 0x2A.

2.3.2 slave (correct)

Address field	Function code 10H	number of bytes	register 1 Hi	register 1 Lo	register 2 Hi	register 2 Lo	.....	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	1 byte		1 byte	1 byte

Example: if address change is succeeded, the correct slave should be '2A 10 00 14 00 02 07 D7'.

2.3.3 slave (incorrect)

Note: Bytes = registers *2

Address field	Function code 90H	Error code	CRC code Lo	CRC code Hi
1 byte	1 byte	1 byte	1 byte	1 byte

2.4 error code

error	Description	Remark
1	illegal function code	The function code is not identified or supported
2	illegal address data	The arrdess data isnot within the range
3	the value is out of range	
4	Checking error	

3 coding sheet

3.1 value data

number	description	unit	data type	data format	Length (bytes)	Start address Hex	Read and (or) write
30001	Voltage of L1 to N	V	int	XXXXXXXX	4	0000	Read only
30003	Voltage of L2 to N	V	int	XXXXXXXX	4	0002	
30005	Voltage of L3 to N	V	int	XXXXXXXX	4	0004	
30007	L1 current	A	int	XXXXXXXX	4	0006	
30009	L2 current	A	int	XXXXXXXX	4	0008	
30011	L3 current	A	int	XXXXXXXX	4	000A	
30013	L1 active power	kW	int	XXXXXXXX	4	000C	
30015	L2 active power	kW	int	XXXXXXXX	4	000E	
30017	L3 active power	kW	int	XXXXXXXX	4	0010	
30031	L1 power factor		int	XXXXXXXX	4	001E	
30033	L2 power factor		int	XXXXXXXX	4	0020	
30035	L3 power factor		int	XXXXXXXX	4	0022	
30053	Total active power	kW	int	XXXXXXXX	4	0034	
30063	Total power factor		int	XXXXXXXX	4	003E	
30073	Total import active energy	kWh	int	XXXXXXXX	4	0048	
30075	Total export active energy	kWh	int	XXXXXXXX	4	004A	
30097	Part import active energy	kWh	int	XXXXXXXX	4	0060	
30099	Part export active energy	kWh	int	XXXXXXXX	4	0062	

Remark:

1. All values except power and power factor are unsigned and have 2 decimals after covert to decimal. Power value is a signed number without decimal (represented with complement). Power factor is a signed number with 3 decimals (represented with complement).
2. Since the data are all 4 bytes with 2 register addresses and the storage order is from high to low. For example, if positive active power is 123456.75 kwh, 1234 is stored in register 0x0048, and 5678 is stored in register 0x0049.

3.2 parameter data

number	description	Data format		length	Start address Hex	read and (or) write
40019	Communication check and stop bit (default is 0) 0: one stop bit and no checking	int	00000000	4	0012	read only
40021	Communication address (1-250)	int	000000NN	4	0014	Readwrite
40029	2-9600bps Baud rate (default is 2) 0 2400bps 1 4800bps 2 9600bps 3 19200bps 5 1200bps	int	00000002	4	001C	Read only
40087	pulse mode (default is 2) 1 reverse active 2 Total active 2-4 positive active	int	00000002	4	0056	
464513	Serial number	BCD	NNNNNNNN	4	FC00	
464515	Meter code	int	0000000D	4	FC02	

4.CRC checking
4.1 Multinomial expression
CRC16 = 1 + x^2 + x^15 + x^16;

4.2 generating programs

```
unsigned short CRC16 ( puchMsg, usDataLen )
* The function returns the CRC as a unsigned short type *
unsigned char *puchMsg ; * message to calculate CRC upon *
unsigned short usDataLen ; * quantity of bytes in message *
{
unsigned char uchCRCHi = 0xFF ; * Hi byte of CRC initialized *
unsigned char uchCRCLo = 0xFF ; * Lo byte of CRC initialized *
unsigned uIndex ; * will index into CRC lookup table *
while (usDataLen—) * pass through message buffer *
{
uIndex = uchCRCLo ^ *puchMsg++ ; * calculate the CRC *
uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
uchCRCHi = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
```

4.3 the table used for CRC calculation
4.3.1 Hi-Order Byte Table

```
* Table of CRC values for Hi–order byte *
static unsigned char auchCRCHi[ ] = {
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
} ;
```

4.3.2 Lo-Order Byte Table

```
* Table of CRC values for Lo–order byte *
static char auchCRCLo[ ] = {
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
};