https://www.lammertbies.nl/comm/info/crc-calculation

<https://www.youtube.com/watch?v=0sDySNuS9dI>[**Microcontrollers**](https://www.youtube.com/@АлександрПисанец)

контрольная сумма CRC + modbus RTU

**Тестовая задача**

С геофизического сенсора приходит пакет байт данных. Необходимо создать функцию для расшифровки пакета данных и проверки CRC используя выдержки из datasheet на оборудование. Вычислить значения содержащиеся в пакете. Ответ прислать в виде готовых функций и результатов вычислений.

**Пакет байт с геофизического сенсора:**

0x80 0x06 0x02 0x03 0x0b 0xf8 0x03 0xdc 0x03 0x7f 0x15 0x8e 0x03 0x57 0x17 0x3f 0x0b 0x04 0x05 0xb9 0xf1 0xb2 0x06 0x04

**Datasheet**

**6.2 SINGLE-PACKET SENDING CONFIGURATION**

**6.2.1 UPLOAD MODE OF COMPONENT DATA**

Upon receipt of 0x80, the DS750 immediately responds by sending binary sensor

component packets (double-byte data, high-byte before) :

< 80 > < 0601 > < 02 > < HX > < GX > < HY > < GY > < HZ > < GZ > < T > < V > <

CRC > < 0604 >

<80> instruction return (that is, the return value of host sending instruction 0x80);

<0601> component data upload synchronization header, if 0x06 appears in the data or

CRC check code, 0x06 will be sent again after 0x06;

<02> component data upload mode;

<HX> is a double byte (x axis magnetic sensor data times 10000);

<GX> is a double byte (x axis acceleration sensor data times 10000);

<HY><GY><HZ><GZ>;same as the component format above;

<T> is a double byte (temperature sensor data multiplied by 100);

<V> is a double byte (input voltage data multiplied by 100);

<CRC> CRC check bit;

<0604> sync tail;

**CRC:**

The data packages structure as below:

80 06 01 02 dataByte [0] ~ dataByte [15] CrcByte[0] CrcByte [1] 06 04

**The sample of the program:**

unsigned short crc=0; ( unsigned short is 16bits data format )

unsigned short crc16tab[256]={

0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,

0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,

0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,

0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,

0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,

0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,

0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,

0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,

0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,

0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,

0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,

0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,

0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,

0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,

0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,

0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,

0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,

0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,

0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,

0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,

0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,

0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,

0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,

0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,

0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,

0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,

0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,

0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,

0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,

0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,

0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,

0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040

};

//CRC generate

void CalcCRC16(unsigned char c)

{

crc = ((unsigned short)((crc >> 8) ^ crc16tab[ (crc ^ c) & 0xFF ]));

}

void CheckCrc( void )

{

crc = 0;

CalcCRC16(0x01);

CalcCRC16(0x02);

for (int i=0;i<16;i++)// all data number

{

CalcCRC16(dataByte [i]);

}

if (CrcByte [0] ==

(unsigned char)(crc>>8) && CrcByte [1] == (unsigned

char)(crc))

{

CrcCheckPass;

}

else

{

CrcCheckFail;

}

}