Simple serial comm. & Calculating CRC16 using C

Asked 4 years, 7 months ago Modified 4 years, 6 months ago Viewed 399 times

0

I have a sensor in which I wish to get some data from. I want to ask it the date or ask it for data.

The sensor uses RS-232; 9600 8N1 to communicate.

The communication packet is composed with a **header**, **payload** and **CRC**. The manual provides the header and payload for whatever you want to do. Each communication packet is composed as follows:

`<SOH> header <STX> payload <ETB> CRC16 <ETX>`

- `<SOH>`: 0x01
- `<STX>`: 0x02
- `<ETB>`: 0x17
- `<ETX>`: 0X03

The manual gives an example if you want to ask for the date, it tells you the header is 0x31 and the payload is 0x41.

Thus the command to send the sensor is: `\x01\x31\x02\x41\x17\CRC16\x03`

Now as an example, the manual *also* calculates the CRC16 for you, and is A0D5 in ASCII. CRC16 needs to be transmitted little endian.

So the full command is now: `\x01\x31\x02\x41\x17\x44\x35\x41\x30\x03`

The manual doesn't provide any other CRC16 calculations, and it expects the user to do it which is fine :)

From the manual: *Each packet is validated by a 16-bit CRC transferred in hexademiical ASCII coded (four chars). CRC is calculated from the header+payload concatenated*

```
WORD CRC16_Compute( BYTE *pBuffer, WORD length )
{
  BYTE i;
  BOOL bs;
  WORD crc=0;

  while( length-- )
```

```
   {
     crc ^= *pBuffer++;
     for( i = 0; i < 8; i++ )
     {
        bs = crc & 1;
        crc >>= 1;
        if( bs )
        {
          crc ^= 0xA001;
        }
     }
   }
 return crc;
 }
```

That is the CRC calculator in C, I am not too savvy with but this code snippet is all they provide and no context.

In ASCII, they are using 1+A (0x31+0x41) to get 2 byte, A0D5. Could someone explain to me what the CRC code is doing, thanks!

c serial-port crc

Share Improve this question Follow

asked Apr 30, 2020 at 22:38
Beaker
67 ● 1 ● 11

That is a pure programming question. The function takes in the data header and payload and calculates a CRC16 checksum, one of the many different variants of CRC16. Did you even research Wikipedia about CRC? "\x44\x35\x41\x30" is "D5A0". "\x01\x31\x02\x41\x17" is fed into CRC code. – Justme Apr 30, 2020 at 22:44

First read the wikipedia article about CRC, *then* ask a focused question about which part you don't understand. You can take a whole course on CRC. – pipe Apr 30, 2020 at 23:10

Certainly, I understand the premise of CRC and bitwise operators. I am getting stuck on the programming - mainly the for loop and i++ – Beaker Apr 30, 2020 at 23:11

A good news; if create an real time application, you never checked CRC. – dsgdfg May 1, 2020 at 0:23

1 Answer

Sorted by: Highest score (default)

1

```
#include <stdio.h>
typedef unsigned char BYTE;
typedef unsigned int BOOL;
typedef unsigned int WORD;
```

```
WORD CRC16_Compute( BYTE *pBuffer, WORD length )
{
  BYTE i;
  BOOL bs;
  WORD crc=0;

  while( length-- )
  {
    crc ^= *pBuffer++;
    for( i = 0; i < 8; i++ )
    {
       bs = crc & 1;
       crc >>= 1;
       if( bs )
       {
         crc ^= 0xA001;
       }
    }
  }
    return crc;
}
int main ( void )
{
    unsigned char data[2];
    unsigned char sdata[9];
    unsigned int x;
    unsigned int z;
    unsigned int i;

    data[0]=0x31;
    data[1]=0x41;
    x = CRC16_Compute(data,2);
    x&=0xFFFF;
    printf("0x%X\n",x);
    z=0;
    sdata[z++]=0x01;
    sdata[z++]=data[0];
    sdata[z++]=0x02;
    sdata[z++]=data[1];
    sdata[z++]=0x17;
    sdata[z  ]=((x>> 4)&0xF)+0x30; if(sdata[4]>0x39) sdata[4]+=7; z++;
    sdata[z  ]=((x>> 0)&0xF)+0x30; if(sdata[5]>0x39) sdata[5]+=7; z++;
    sdata[z  ]=((x>>12)&0xF)+0x30; if(sdata[6]>0x39) sdata[6]+=7; z++;
    sdata[z  ]=((x>> 8)&0xF)+0x30; if(sdata[7]>0x39) sdata[7]+=7; z++;
    sdata[z++]=0x03;
    for(i=0;i<z;i++) printf("%02X ",sdata[i]); printf("\n");
    return(0);
}
```

Run it

```
gcc so.c -o so
./so
0xA0D5
01 31 02 41 17 44 35 41 30 03
```

How about that the right answer....

Everything you need to know is in your question, just do what it says. A few minutes of crudely putting it together.

> CRC is calculated from the header+payload concatenated

```
data[0]=0x31;
data[1]=0x41;
```

That is header and payload and it gives the right answer based on the CRC code provided.

Then you build the packet with the other items. If you google ASCII table you can see the values for 'D' 'A' '0' '5' and can figure out how to get from 0xD to 0x44 and 0xA to 0x41 but first look at 0x0 to 0x30 and 0x5 to 0x35, 0-9 is easy but 0x0A gives 0x3A but needs to be 0x41, so you adjust.

So the code works as described based on the comments as described, I don't know this sensor, seems goofy the way they did it but good for them for providing an example and the details on the crc16 as there are multiple standard variations including the initial value, so again good for them for saving tons of time trying to figure it out...

Share

Improve this answer

Follow

edited Jun 9, 2020 at 20:18
halfer
20.4k ● 19 ● 108 ● 200

answered May 1, 2020 at 0:09
old_timer
71.4k ● 9 ● 98 ● 174

Cut throat out here, eh? Thanks, I appreciate the time you took to formulate a response.
– Beaker May 1, 2020 at 1:42

A bit more on old-timer's comment that " there are multiple standard variations including the initial value". Besides knowing the CRC polynomial being used, in order to get consistent results, you also need to specify 1) the initial value for the CRC register, 2) should the CRC be complimented (XOR'd) before being appended to the message, 3) the order in which the CRC bytes are appended to the message (little endian vs big endian), and 4) whether the bits within each byte are reflected or not, which some hardware does and others do not.
– SteveSh May 1, 2020 at 18:58

This text might provide some insight: "A Painless Guide to CRC Error Detection Algorithms"
– the busybee Jun 8, 2020 at 7:00