一起学 CC3200 之

CRC 校验

小阿汤哥

序：
能力有限，难免有错，有问题请联系我，请留言或者邮件联系
QQ 群交流：482729453　邮件联系 hytga@163.com
资料共享链接 http://pan.baidu.com/s/1hqiWB56
版本：20160323

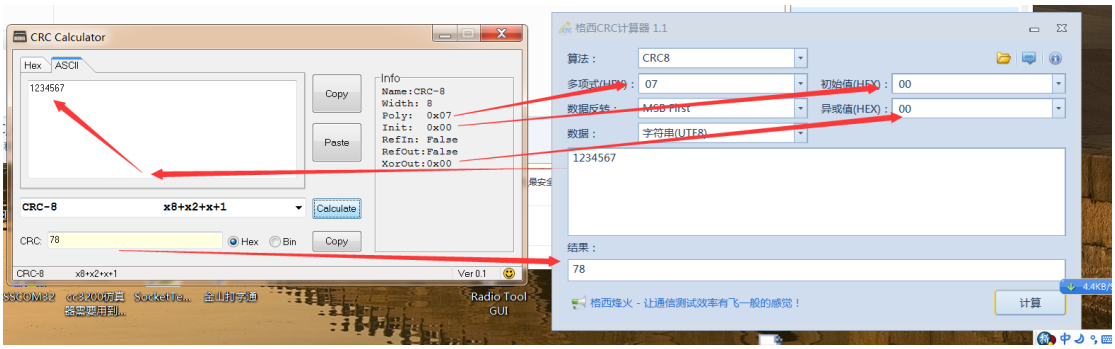
一、 CRC 简介

百度百科：CRC 即循环冗余校验码（Cyclic Redundancy Check[1] ）：是数据通信领域中最常用的一种查错校验码，其特征是信息字段和校验字段的长度可以任意选定。循环冗余检查（CRC）是一种数据传输检错功能，对数据进行多项式计算，并将得到的结果附在帧的后面，接收设备也执行类似的算法，以保证数据传输的正确性和完整性。

那么什么是 CRC？简单地说就是一个用来判断数据有没有错，譬如：数据：“432432”的检验码为 90，你把“432432”和 90 发给其他设备，其他设备对“432432”进行计算，如果得到的值不是 90，那么这串数据就是有问题的。有问题的数据肯定是直接扔掉的。

CRC 有一个比较重要的东西叫做多项式：多项式为 $x^5+x^3+x^2+x+1$ 对应的代码 101111。这个具体是做啥的？自己百度下。

我有两个软件请看：两个字符串计算出来的是一样的。



CRC 的多项式有很多种：

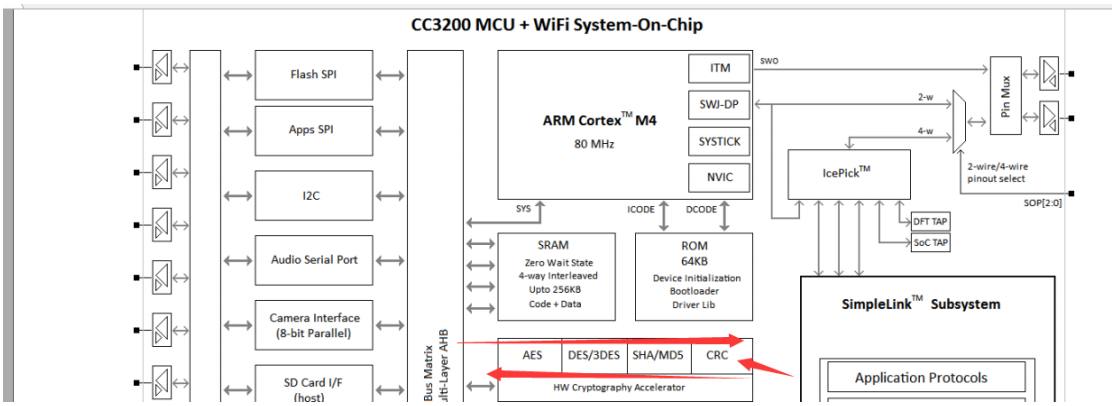
格西CRC标准表

名称	多项式	初始值	异或值	Bit反转
CRC-8	07	00	00	MSB First
CRC-8/DARC	39	00	00	LSB First
CRC-8/ITU	07	00	55	MSB First
CRC-8/MAXIM	31	00	00	LSB First
CRC-8/ROHC	07	FF	00	LSB First
CRC-16	8005	0000	0000	LSB First
CRC-16/A	1021	C6C6	0000	LSB First
CRC-16/CCITT	1021	0000	0000	LSB First
CRC-16/CCITT-FALSE	1021	FFFF	0000	MSB First
CRC-16/DECT R	0589	0000	0001	MSB First
CRC-16/DECT X	0589	0000	0000	MSB First
CRC-16/DNP	3D65	0000	FFFF	LSB First
CRC-16/GENIBUS	1021	FFFF	FFFF	MSB First
CRC-16/MAXIM	8005	0000	FFFF	LSB First
CRC-16/MODBUS	8005	FFFF	0000	LSB First
CRC-16/USB	8005	FFFF	FFFF	LSB First
CRC-16/X25	1021	FFFF	FFFF	LSB First
CRC-16/XMODEM	1021	0000	0000	MSB First
CRC-32	04C11DB7	FFFFFFFF	FFFFFFFF	LSB First
CRC-32/B	04C11DB7	FFFFFFFF	FFFFFFFF	MSB First
CRC-32/C	1EDC6F41	FFFFFFFF	FFFFFFFF	LSB First
CRC-32/D	A833982B	FFFFFFFF	FFFFFFFF	LSB First
CRC-32/MPEG-2	04C11DB7	FFFFFFFF	00000000	MSB First
CRC-32/POSIX	04C11DB7	00000000	FFFFFFFF	MSB First

确定 取消

二、 CC3200 的 CRC

CC3200 内部具有硬件的 CRC 校验功能。



目前找不到有关 CRC 模块的介绍，只能从官方例程找到怎么用这个 CRC，可能是官方觉得这个太简单了，没必要写了吧。

但是我告诉你 CRC 其实是有 bug 的。为什么说是有 bug，请继续看。当然如果对于懂的人来说可能不是 bug。对于不懂的人来说，想了老半天，不知道是啥意思？毕竟没有中文文档，除非你去问官方。

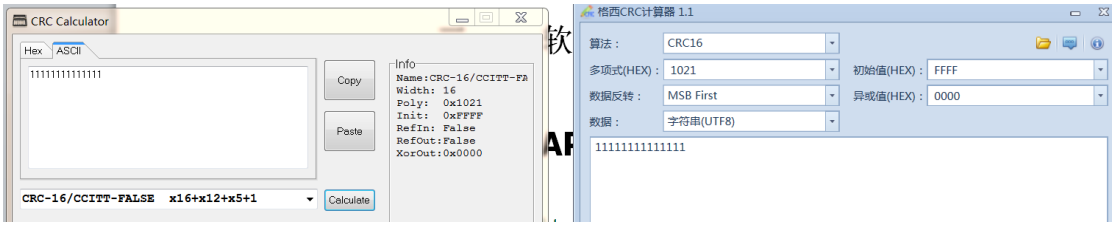
CC3200 只支持一部分的多项式：5 种

```
70 #define CRC_CFG_TYPE_P8005     0x00000000  // Polynomial 0x8005
71 #define CRC_CFG_TYPE_P1021     0x00000001  // Polynomial 0x1021
72 #define CRC_CFG_TYPE_P4C11DB7  0x00000002  // Polynomial 0x4C11DB7
73 #define CRC_CFG_TYPE_P1EDC6F41 0x00000003  // Polynomial 0x1EDC6F41
74 #define CRC_CFG_TYPE_TCPCHKSUM 0x00000008  // TCP checksum
```

够用了，如果不支持的，可以去找一些代码（肯定是有人奉献出来了），或者自己编写。

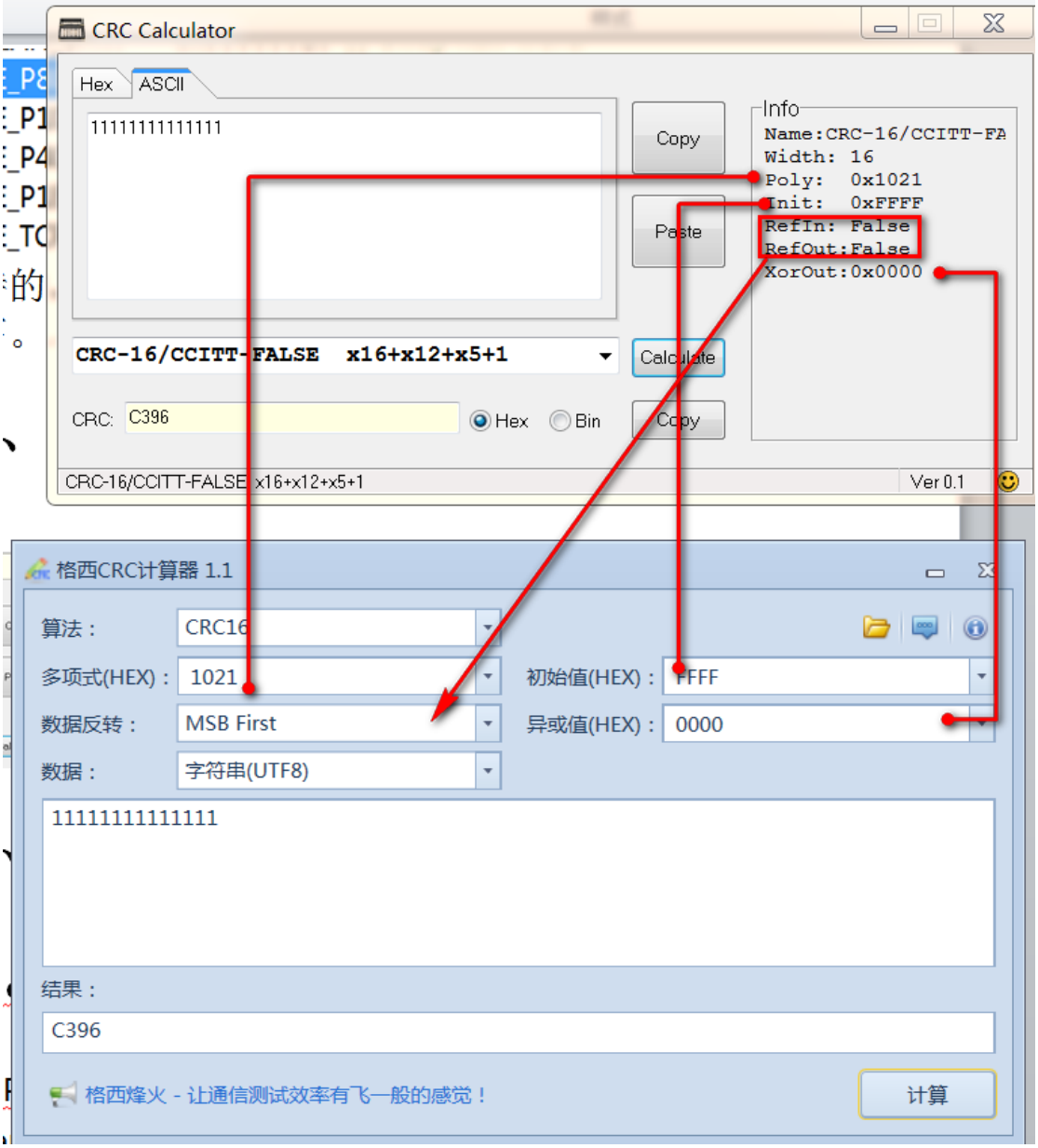
三、 软件说明

先说说这两个工具



上面两图的信息是有点不太一样的，我是没研究 CRC，反正这东西拿过就是验证下对不对，然后下了两个软件，结果发现有点不太一样，不同人写的，估计就是不太一样的，那么怎么把两个软件一一对应起来？为什么要对应起来？只因为我不知道哪个是正确的。

请看下图，红色连起来就是有直接关系的。

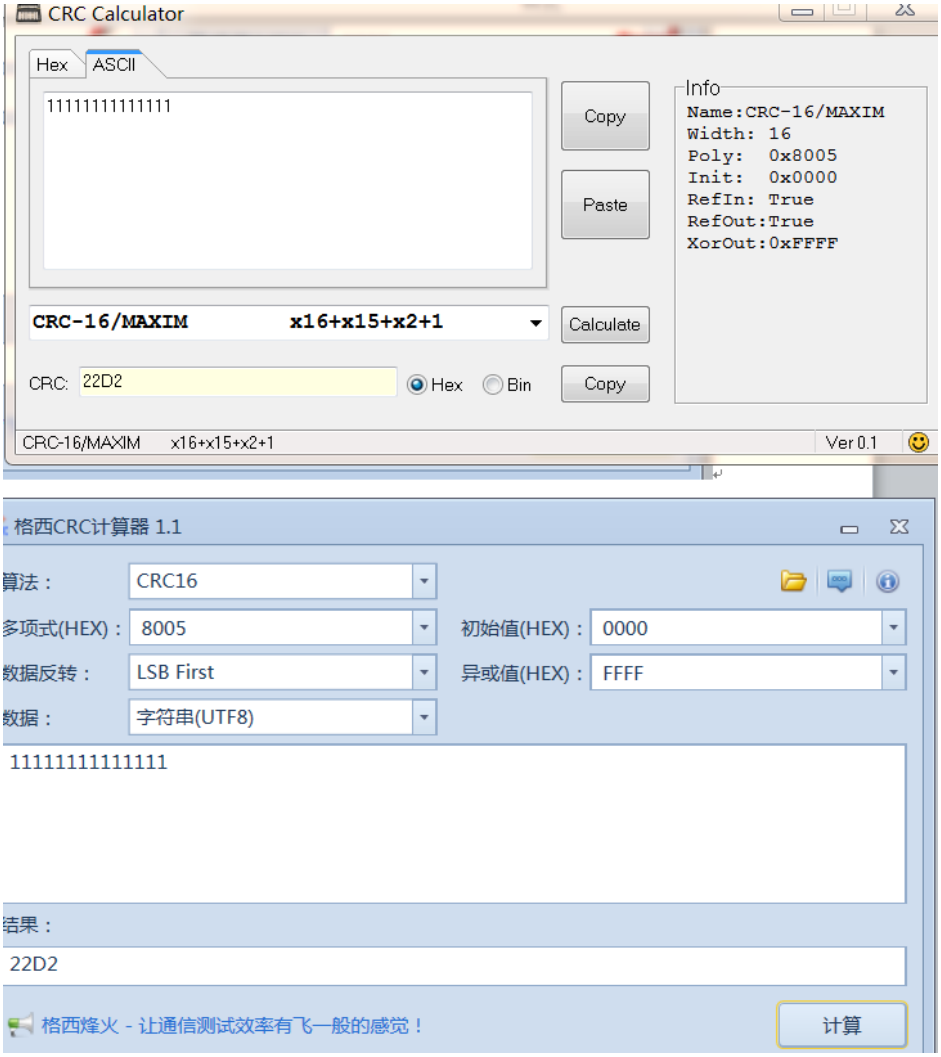


CRC Calculator	格西CRC计算器 1.1
Ploy	多项式
Init	初始值
Refin==false refou==false	数据反转：MSB first
Refin==True refou==true	数据反转：LSB first
Xorout	异或值

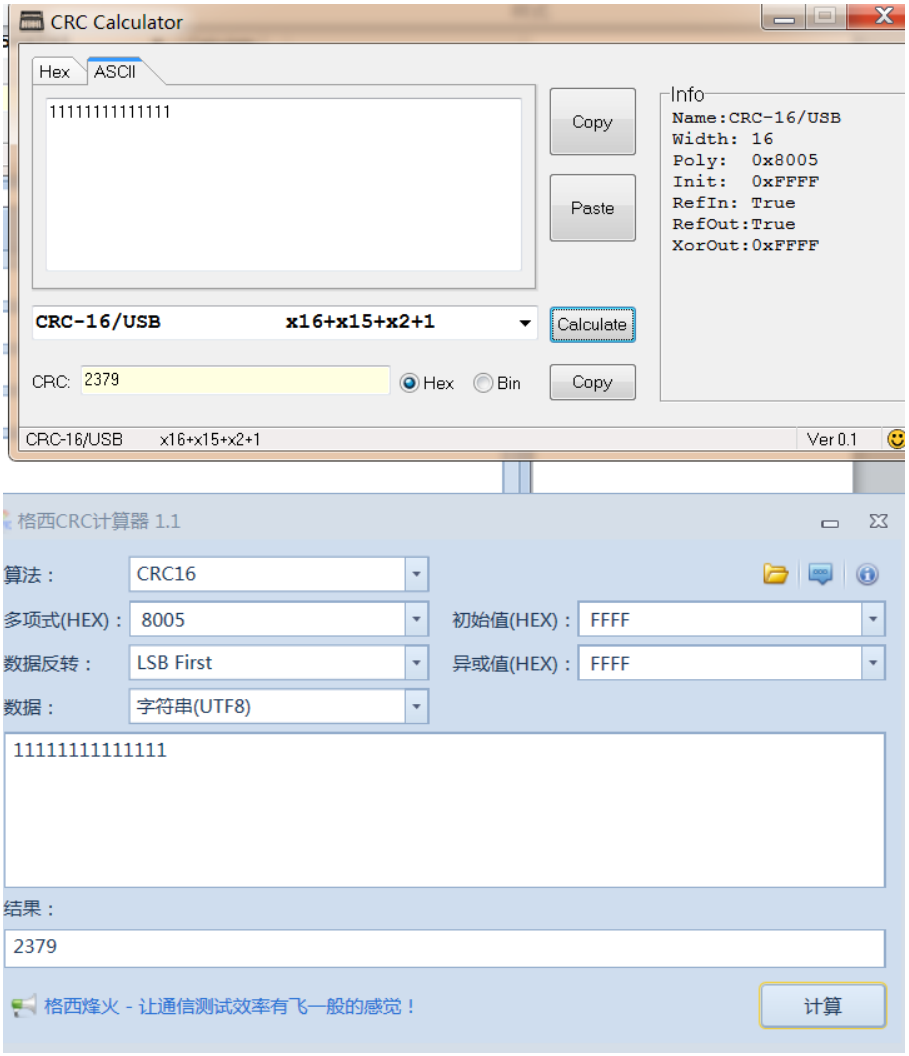
请注意上面是根据经验得到的，他们为什么这样我并不知道。没去研究 CRC 算法。

下面有这两个的测试示例：

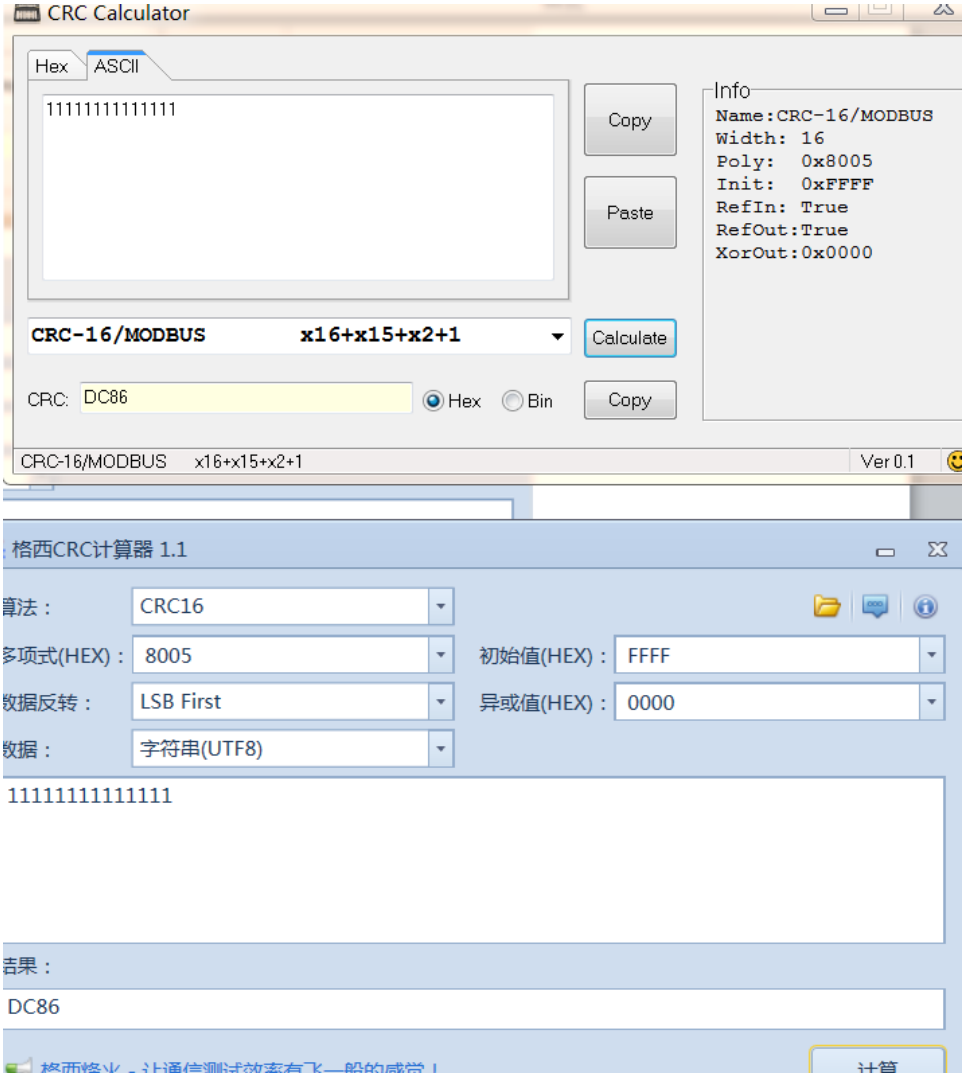
测试 1：


CRC Calculator
Hex
ASCII
11111111111111
Copy
Paste
Info
Name:CRC-16/MAXIM
Width: 16
Poly: 0x8005
Init: 0x0000
RefIn: True
RefOut:True
XorOut:0xFFFF
CRC-16/MAXIM x16+x15+x2+1
Calculate
CRC: 22D2
Hex
Bin
Copy
CRC-16/MAXIM x16+x15+x2+1
Ver 0.1
格西CRC计算器 1.1
算法：
CRC16
多项式(HEX)：
8005
初始值(HEX)：
0000
数据反转：
LSB First
异或值(HEX)：
FFFF
数据：
字符串(UTF8)
11111111111111
结果：
22D2
格西烽火 - 让通信测试效率有飞一般的感觉！
计算

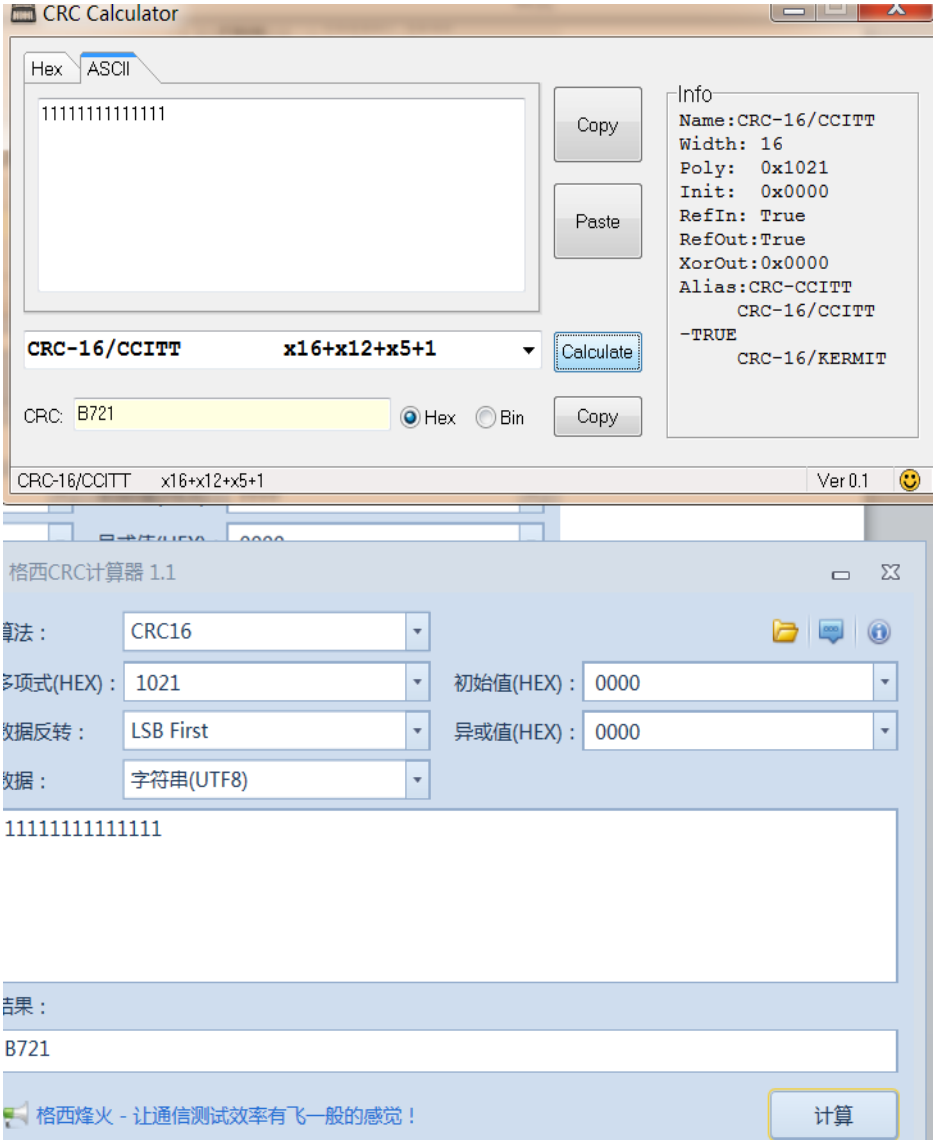
测试 2:


CRC Calculator
Hex
ASCII
11111111111111
Copy
Paste
Info
Name:CRC-16/USB
Width: 16
Poly: 0x8005
Init: 0xFFFF
RefIn: True
RefOut:True
XorOut:0xFFFF
CRC-16/USB x16+x15+x2+1
Calculate
CRC: 2379
Hex
Bin
Copy
CRC-16/USB x16+x15+x2+1
Ver 0.1
格西CRC计算器 1.1
算法：
CRC16
多项式(HEX)：
8005
初始值(HEX)：
FFFF
数据反转：
LSB First
异或值(HEX)：
FFFF
数据：
字符串(UTF8)
11111111111111
结果：
2379
格西烽火 - 让通信测试效率有飞一般的感觉！
计算

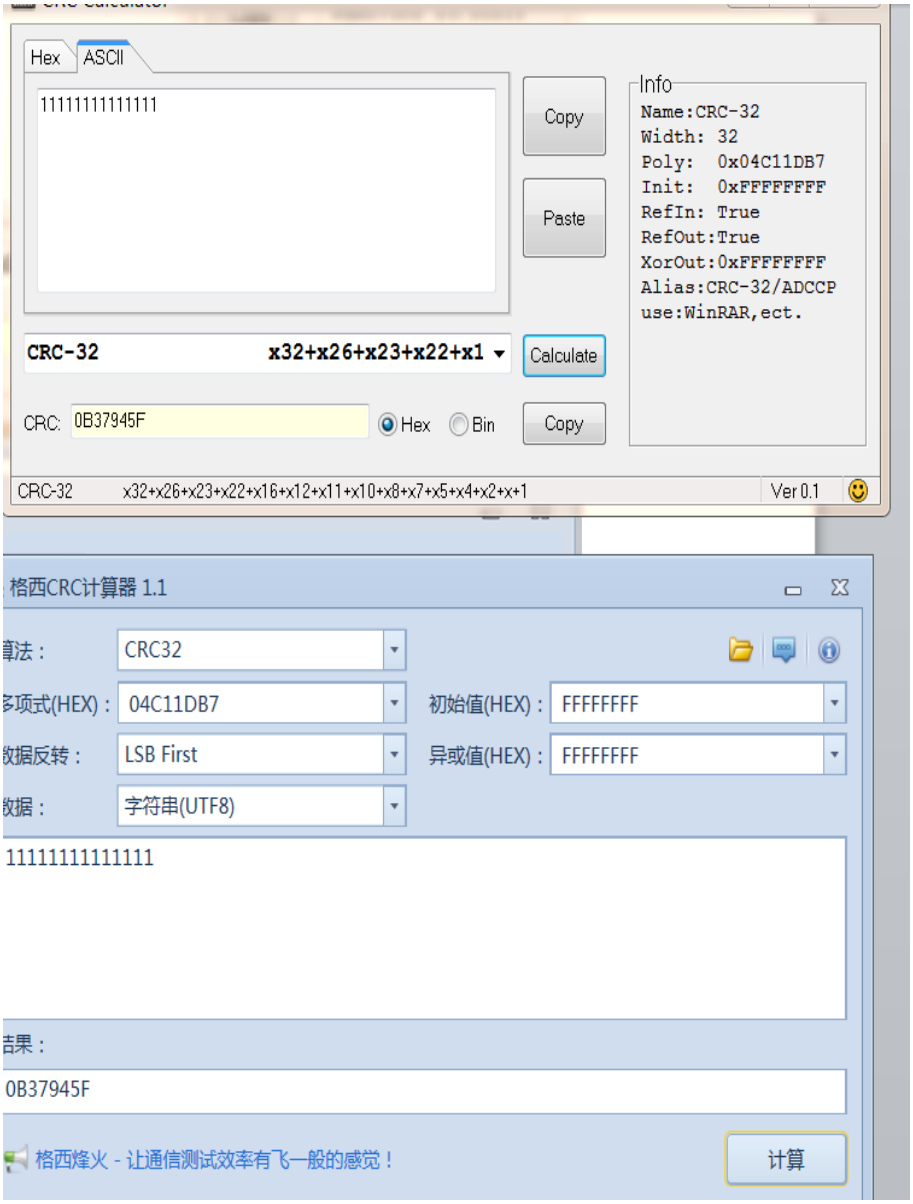
测试 3：


CRC Calculator
Hex
ASCII
11111111111111
Copy
Paste
Info
Name:CRC-16/MODBUS
Width: 16
Poly: 0x8005
Init: 0xFFFF
RefIn: True
RefOut:True
XorOut:0x0000
CRC-16/MODBUS x16+x15+x2+1
Calculate
CRC: DC86
Hex
Bin
Copy
CRC-16/MODBUS x16+x15+x2+1
Ver 0.1
格西CRC计算器 1.1
算法：
CRC16
多项式(HEX)：
8005
初始值(HEX)：
FFFF
数据反转：
LSB First
异或值(HEX)：
0000
数据：
字符串(UTF8)
11111111111111
结果：
DC86

测试 4：


CRC Calculator
Hex
ASCII
11111111111111
Copy
Paste
Info
Name:CRC-16/CCITT
Width: 16
Poly: 0x1021
Init: 0x0000
RefIn: True
RefOut:True
XorOut:0x0000
Alias:CRC-CCITT
CRC-16/CCITT
-TRUE
CRC-16/KERMIT
CRC-16/CCITT x16+x12+x5+1
Calculate
CRC: B721
Hex
Bin
Copy
CRC-16/CCITT x16+x12+x5+1
Ver 0.1
格西CRC计算器 1.1
算法：
CRC16
多项式(HEX)：
1021
初始值(HEX)：
0000
数据反转：
LSB First
异或值(HEX)：
0000
数据：
字符串(UTF8)
11111111111111
结果：
B721
格西烽火 - 让通信测试效率有飞一般的感觉！
计算

测试 5：



四、 API 函数

//配置CRC

1. **extern void CRCConfigSet**(uint32_t ui32Base, uint32_t ui32CRCConfig);

//数据处理

2. **extern** uint32_t CRCDataProcess(uint32_t ui32Base, **void** *puiDataIn,
   uint32_t ui32DataLength, uint32_t ui32Config);

//写数据

3. **extern void CRCDataWrite**(uint32_t ui32Base, uint32_t ui32Data);

//读结果

4. **extern** uint32_t **CRCResultRead**(uint32_t ui32Base);

//设置种子

5. **extern void CRCSeedSet**(uint32_t ui32Base, uint32_t ui32Seed);

这上面序号为 1 ， 2 ，5 比较常用，另外两个没啥作用，CRCDataProcess 这个函数就是包含了 CRCDataWrite 和 CRCResultRead 这两个功能。

五、 测试及示例代码

CRC 很简单：

1. 使能时钟。
2. 配置。
3. 输入数据返回结果。

简单吧！

示例：

多项式为：

CRC-16	8005	0000	0000	LSB First

数据宽度为：8bit

种子为：0

什么是种子？假设你要上传 10000T 的数据到网盘，怎么办？只能注册 10000 个账号，一个账号传 1T，哈哈，撑死百度。

这种子也是这个作用，你检验 1000 个数据，分批校验，第一次检验 100，得到结果假设为 90，那么 90 就作为种子，进行下一次的校验。就这样一直循环，直到校验完成。

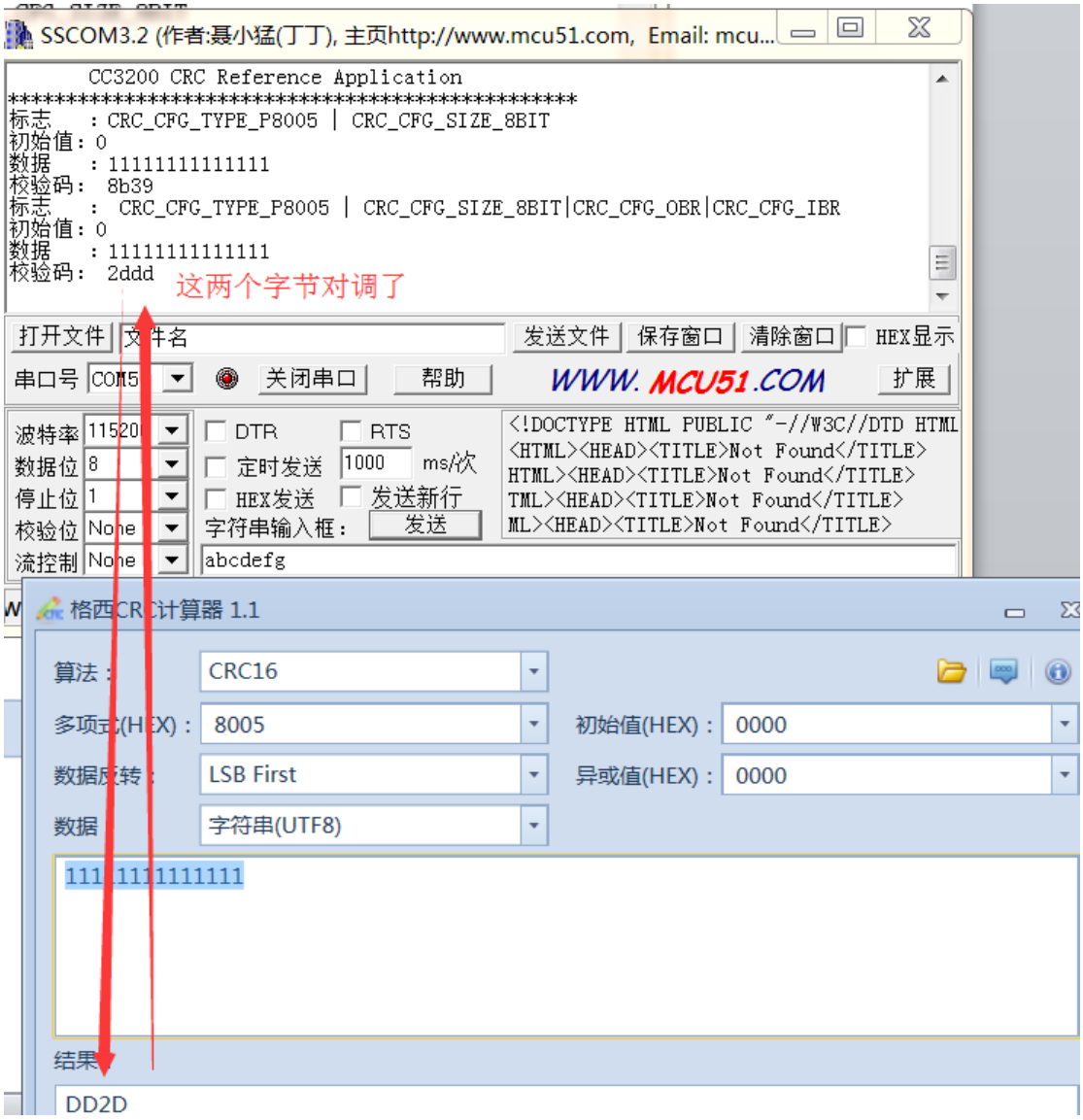
什么是数据宽度？数据宽度就是你输入的数据的最小宽度是 8bit 或者是 32bit。

示例：代码将放在后面

测试 1

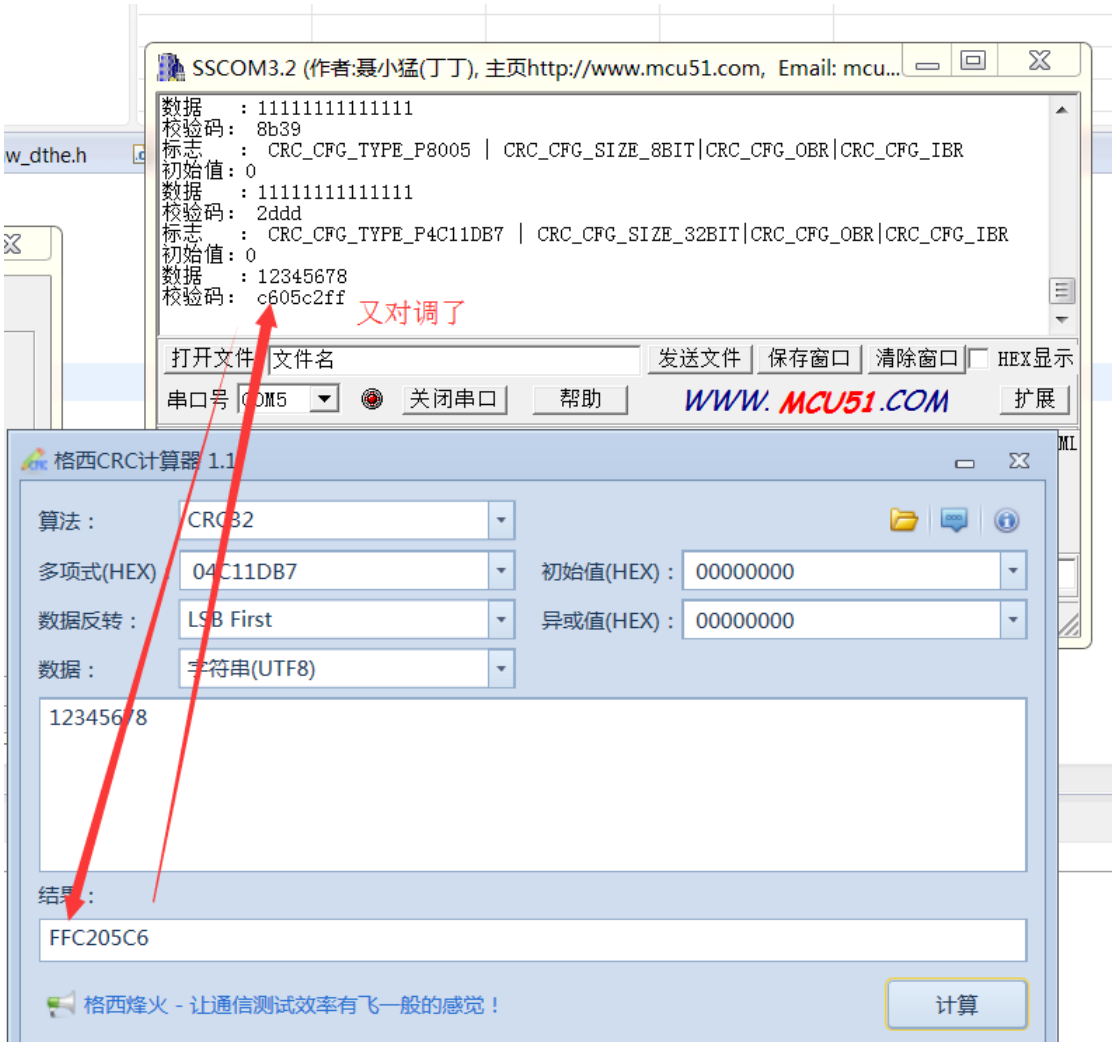

SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com, Email: mcu...
****
CC3200 CRC Reference Application
****
标志 : CRC_CFG_TYPE_P8005 | CRC_CFG_SIZE_8BIT
初始值: 0
数据 : 11111111111111
校验码: 8b39
格西CRC计算器 1.1
算法：
CRC16
多项式(HEX)：
8005
初始值(HEX)：
0000
数据反转：
MSB First
异或值(HEX)：
0000
数据：
字符串(UTF8)
11111111111111
结果：
8B39

测试 2

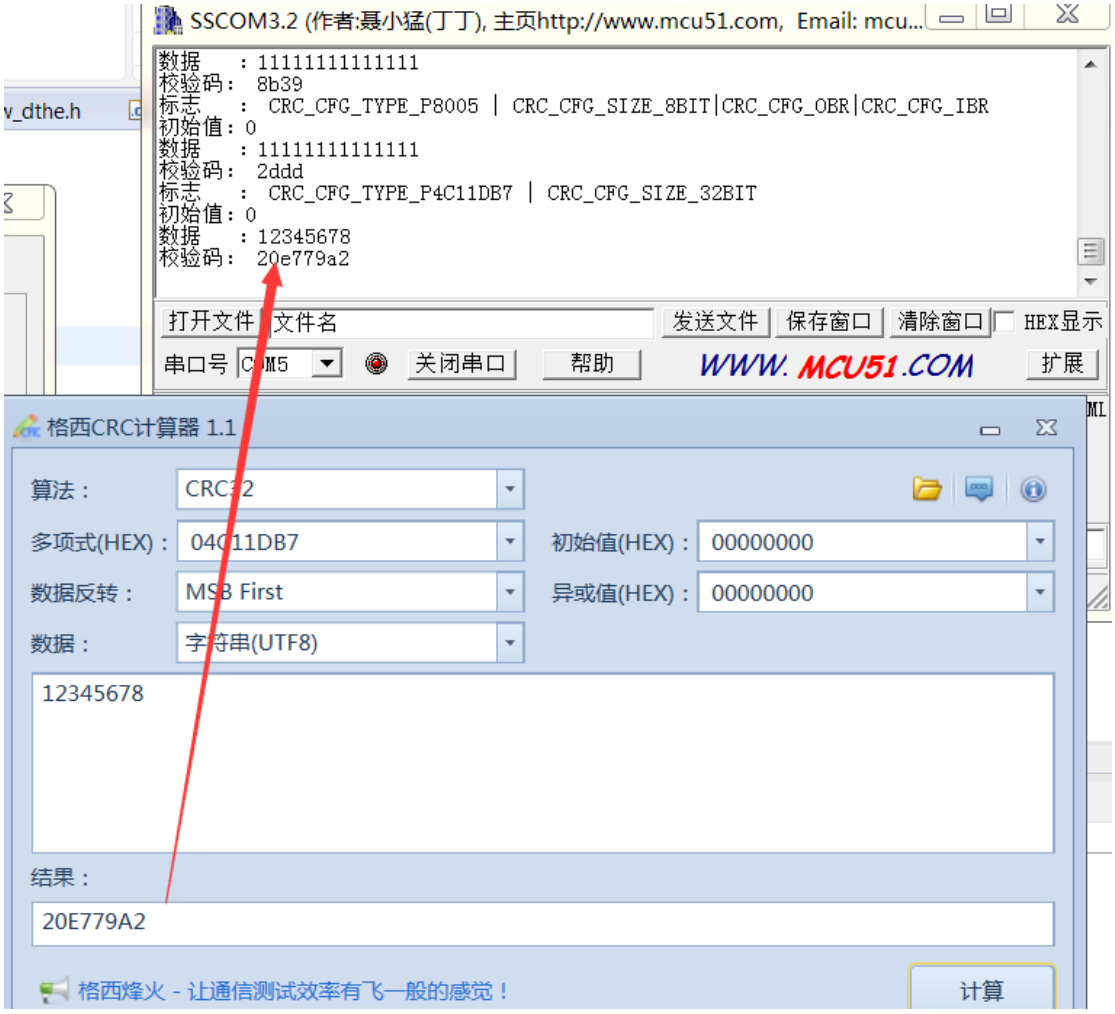


这个就是 bug 所在，这个需要对调字节。

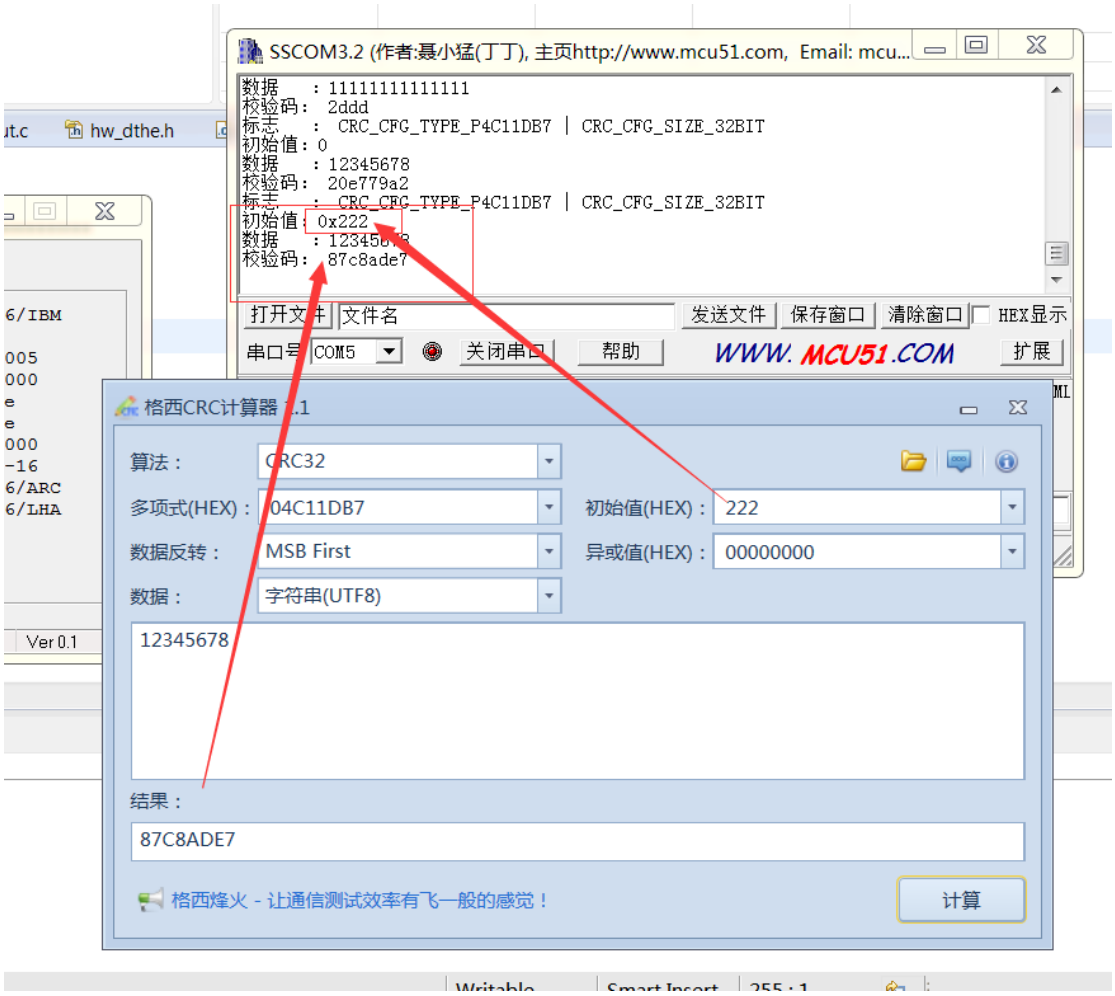
测试 3.


SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com, Email: mcu...
数据 : 11111111111111
校验码: 8b39
标志 : CRC_CFG_TYPE_P8005 | CRC_CFG_SIZE_8BIT|CRC_CFG_OBR|CRC_CFG_IBR
初始值: 0
数据 : 11111111111111
校验码: 2ddd
标志 : CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT|CRC_CFG_OBR|CRC_CFG_IBR
初始值: 0
数据 : 12345678
校验码: c605c2ff
又对调了
打开文件
文件名
发送文件
保存窗口
清除窗口
HEX显示
串口号
COM5
关闭串口
帮助
WWW.MCU51.COM
扩展
格西CRC计算器 1.1
算法：
CRC32
多项式(HEX)：
04C11DB7
初始值(HEX)：
00000000
数据反转：
LSB First
异或值(HEX)：
00000000
数据：
字符串(UTF8)
12345678
结果：
FFC205C6
格西烽火 - 让通信测试效率有飞一般的感觉！
计算

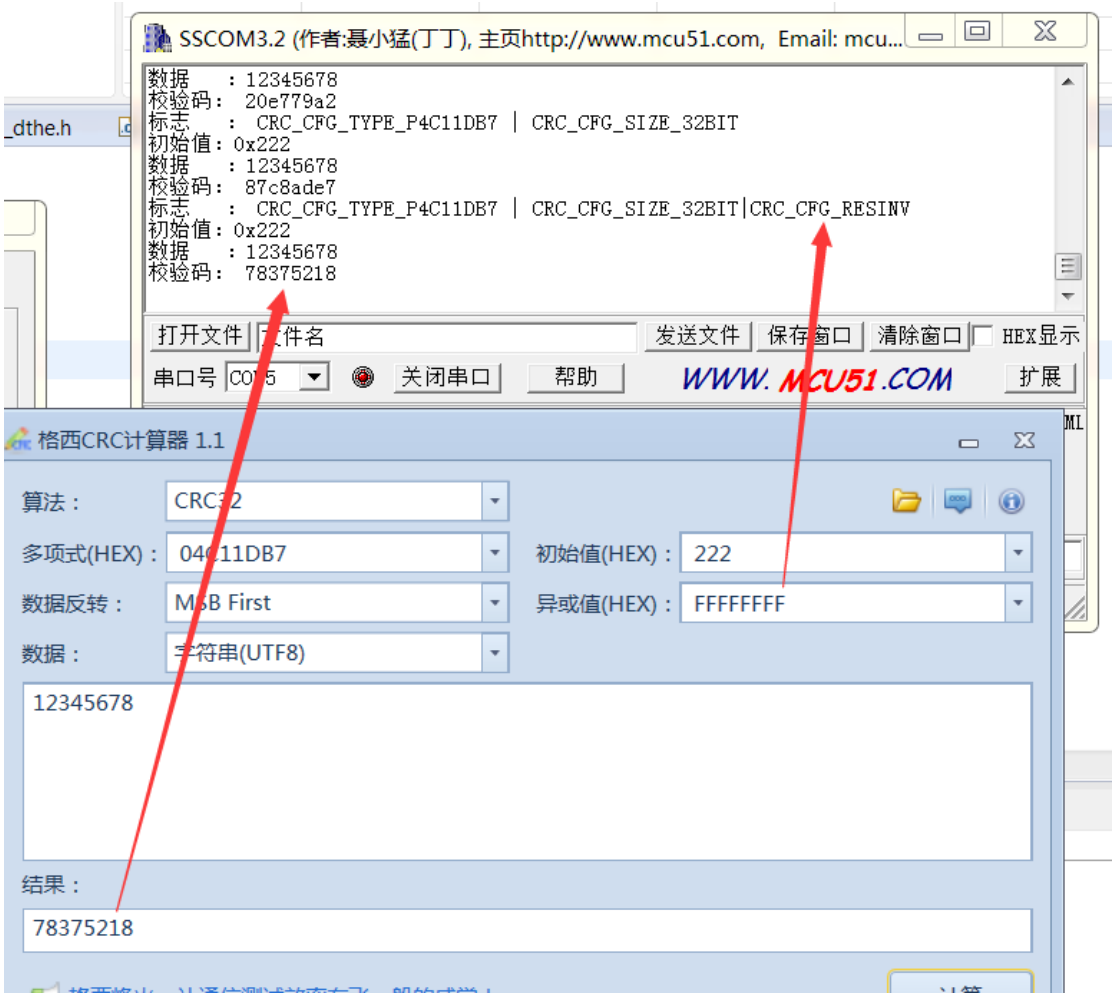
测试 4


SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com, Email: mcu...
数据 : 11111111111111
校验码: 8b39
标志 : CRC_CFG_TYPE_P8005 | CRC_CFG_SIZE_8BIT|CRC_CFG_OBR|CRC_CFG_IBR
初始值: 0
数据 : 11111111111111
校验码: 2ddd
标志 : CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT
初始值: 0
数据 : 12345678
校验码: 20e779a2
打开文件
文件名
发送文件
保存窗口
清除窗口
HEX显示
串口号
关闭串口
帮助
WWW.MCU51.COM
扩展
格西CRC计算器 1.1
算法:
多项式(HEX):
04C11DB7
初始值(HEX):
00000000
数据反转:
MSB First
异或值(HEX):
00000000
数据:
字符串(UTF8)
12345678
结果:
20E779A2
格西烽火 - 让通信测试效率有飞一般的感觉!
计算

测试 5


SSCOM3.2 (作者:聂小猛(丁丁), 主页http://www.mcu51.com, Email: mcu...
数据 : 11111111111111
校验码: 2ddd
标志 : CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT
初始值: 0
数据 : 12345678
校验码: 20e779a2
标志 : CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT
初始值: 0x222
数据 : 12345678
校验码: 87c8ade7
打开文件
文件名
发送文件
保存窗口
清除窗口
HEX显示
串口号
COM5
关闭串口
帮助
WWW.MCU51.COM
扩展
格西CRC计算器 1.1
算法：
CRC32
多项式(HEX)：
04C11DB7
初始值(HEX)：
222
数据反转：
MSB First
异或值(HEX)：
00000000
数据：
字符串(UTF8)
12345678
结果：
87C8ADE7
格西烽火 - 让通信测试效率有飞一般的感觉！
计算

测试 6



代码

```
MAP_PRCMPeripheralClkEnable(PRCM_DTHE, PRCM_RUN_MODE_CLK);
    unsigned long  retVal;
    //配置 1
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P8005 | CRC_CFG_SIZE_8BIT);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "11111111111111",
                               sizeof("11111111111111")  - 1, CRC_CFG_SIZE_8BIT);
    UART_PRINT("标志    : CRC_CFG_TYPE_P8005 | CRC_CFG_SIZE_8BIT \r\n");
    UART_PRINT("初始值：0 \r\n");
    UART_PRINT("数据    : 11111111111111   \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);
```

```
    //配置 2
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P8005 |
CRC_CFG_SIZE_8BIT|CRC_CFG_OBR|CRC_CFG_IBR);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "11111111111111",
                               sizeof("11111111111111")  - 1, CRC_CFG_SIZE_8BIT);
    UART_PRINT("标志    ：CRC_CFG_TYPE_P8005 |
CRC_CFG_SIZE_8BIT|CRC_CFG_OBR|CRC_CFG_IBR \r\n");
    UART_PRINT("初始值：0 \r\n");
    UART_PRINT("数据    ：11111111111111   \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);


    //配置 3
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P4C11DB7 |
CRC_CFG_SIZE_32BIT|CRC_CFG_OBR|CRC_CFG_IBR);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "12345678",
                               2, CRC_CFG_SIZE_32BIT);
    UART_PRINT("标志    ：CRC_CFG_TYPE_P4C11DB7 |
CRC_CFG_SIZE_32BIT|CRC_CFG_OBR|CRC_CFG_IBR \r\n");
    UART_PRINT("初始值：0 \r\n");
    UART_PRINT("数据    ：12345678   \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);

    //配置 4
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "12345678",
                               2, CRC_CFG_SIZE_32BIT);
    UART_PRINT("标志    ：CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT \r\n");
    UART_PRINT("初始值：0 \r\n");
    UART_PRINT("数据    ：12345678   \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);
```

```
    //配置 5
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0x222);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "12345678",
                               2, CRC_CFG_SIZE_32BIT);
    UART_PRINT("标志   ： CRC_CFG_TYPE_P4C11DB7 | CRC_CFG_SIZE_32BIT \r\n");
    UART_PRINT("初始值：0x222 \r\n");
    UART_PRINT("数据   ：12345678  \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);



    //配置 6
    MAP_CRCConfigSet(CCM0_BASE, CRC_CFG_TYPE_P4C11DB7 |
CRC_CFG_SIZE_32BIT|CRC_CFG_RESINV);
    //种子
    MAP_CRCSeedSet(CCM0_BASE, 0x222);
    //数据输入，得到结果
    retVal= MAP_CRCDataProcess(CCM0_BASE, (void*) "12345678",
                               2, CRC_CFG_SIZE_32BIT);
    UART_PRINT("标志   ： CRC_CFG_TYPE_P4C11DB7 |
CRC_CFG_SIZE_32BIT|CRC_CFG_RESINV \r\n");
    UART_PRINT("初始值：0x222 \r\n");
    UART_PRINT("数据   ：12345678  \r\n");
    UART_PRINT("校验码： %x\r\n", retVal);
```

六、 总结

CC3200 的 CRC 有 bug，或者是我少设置了其他东西，具体看第五部分测试及示例代码。