查看: 47654 | 回复: 258

分享我的项目必需品：IAP+YMODEM+CRC16+AES256+PC端软件+hex合并 [复制链接]

it_do_just

14 主题 784 帖子 5 精华

论坛大神

积分 2997
金钱 2997
注册时间 2015-7-26
在线时间 770 小时

发消息

发表于 2016-7-3 18:26:53 | 只看该作者 | 只看大图 楼主 电梯直达

本帖最后由 it_do_just 于 2019-8-19 08:42 编辑

由于项目需要，花时间研究了一下有关IAP的知识，虽然在原子哥的教程中有讲到关于IAP，但是实际项目中并不会那样使用，也不会使用串口直接不通过协议传输文件，但是原子哥的教程很好的讲明白了IAP的思路以及实现的方法，至于细节部分就是我开贴的原因，希望能多把这种实际项目一定会用到的知识为大家所知，一下代码使用的平台是STM32F0，改成F1/F4也很简单，可根据自己情况修改

IAP：

学习IAP之前建议大家先看看原子哥的教程了解一下大概，知道IAP的作用以及实现，下面提供一个网址(http://www.openedv.com/posts/list/11494.htm) ，是原子哥的IAP文字教程，看完后起码按照教程顺序下来实现串口升级代码了，但是教程中有个跳转至中断向量表的知识点我觉得可能存在问题，我也发帖询问过，但是没有得到解答，以下是我的提问帖，希望能得到答复
(http://www.openedv.com/forum.php?mod=viewthread&tid=75843)

YMODEM：

Ymodem的知识简介自己百度一下吧，传送门 (http://baike.baidu.com/link?url=Z7sLoTcGJjKH580EUmInqTFIZnPYUM4IH-Tj-TMYVOy7vOmp7L_J5E5ADX8O97rHLvjX-AVM6LAPkslPUvV6qK)，原子哥采用的是串口直传升级文件，显然在实际项目我们一般不会那么做，因为可能传输出现错误，需要采取些容错、重发和校验等一些措施来避免传输错误，原本有考虑自己写这个协议来实现文件传输，但是上ST官网一查，已经有了现成的YMODEM+CRC16的代码，所以没多想，直接移植过来好了，也许就是所谓的傻瓜式编程了。但是我觉得去了解YMODEM协议是如何实现的对自己技术水平的提高有很大的帮助，在遇到移植出错时起码知道从哪里下手修改代码，官方找来的IAP+Ymodem的代码在附件中，需要的可以自行下载查看。这里主要讲解一下几个重要的函数，其中主要有menu.c /ymodem.c / flash_if.c / common.c 这几个头文件，menu.c主要实现的是调用ymodem协议的接口然后对传输是否成功以及传输文件的大小名字做一些显示，最关键还是ymodem.c，实现通过ymodem协议发送和接收文件，其中作为接收一方，我们只用到"Ymodem_Receive(uint8_t *buf)"这个函数，其中buf是将接收到的1024字节通过ymodem协议接收下来存到缓存buf里面，每传送完成1024字节就写入一次FLASH，这样只需要开辟一个1024字节的缓存就可以实现升级了，而不需要开辟一个很大的数组存储所有代码，没必要浪费那么大一片存储空间，这就是边写边存的好处，在源码中flash_if.c主要就是对flash的操作了，里面有个FLASH_If_Write函数很好的实现了这个功能，而common.c里面就是现实些整数和字符串的转换和串口发送。把这个协议移植到自己代码中的时候还是跳了不少坑的，官方的东西有时候也不一定靠谱，我发现我ymodem接收数据后写入flash的时候总是写入错误，写入flash指定地址后从同个地址读出那个值做二次判断的时候居然不同，我使用的是电脑的超级终端，win7网上可以随便下载得到，XP有自带。这个软件我也会添加到下面的附件当中，里面集成了各种文件传输协议，其中就有ymodem，只要将代码生成bin文件后直接通过ymodem发送出去，然后串口收到的数据通过官方的这段代码就直接能用了。回到话题，写入flash总是出错，后面经过debug发现官方在写flash之前居然不解锁和上锁！不解锁和上锁！！！后面我在写入前加了解锁，写入后上锁，写入错误这个bug就没了，但是还有个问题官方处理的不太严谨， 就是代码文件是否写入完成不是按照文件的大小来判断的，而是通过设定的代码区ash大小来决定的，所以我修改了一些代码，在往flash写入数据的时候判断是否已经把整个文件写完了，由于ymodem协议传输文件时首先会传递文件的大小的一些信息，所以这个文件的大小通过第一个包就可以获取到了，修改的是"FLASH_If_Write"这个函数，可以对比查看，另外官方在写入前会将要写入的那个flash地址先擦除一

遍，但是也没有先解锁再上锁的操作，这个地方我也加上了，我会将我修改后ymodem协议添加到附件中，官方的我也会给在附件中，需要的可以自己查看
"Ymodem_Receive (uint8_t *buf)"这个函数的区别。

在官方例程中还需要修改一个串口发送和接收函数(在commo.h中)，官方的代码的串口发送这样的：

```
01.  void SerialPutChar(uint8_t c)
02.  {
03.    USART_SendData(EVAL_COM1, c);
04.
05.    while (USART_GetFlagStatus(EVAL_COM1, USART_FLAG_TXE) == RESET)
06.    {}
07.  }
08.
09.  //改成如下：串口几可以根据自己的情况修改
10.  void SerialPutChar(uint8_t c)
11.  {
12.    USART_SendData(USART1, c);
13.
14.    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
15.    {}
16.  }
17.
18.  //官方的串口接收如下：
19.  uint32_t SerialKeyPressed(uint8_t *key)
20.  {
21.    if ( USART_GetFlagStatus(EVAL_COM1, USART_FLAG_RXNE) != RESET)
22.    {
23.      *key = USART_ReceiveData(EVAL_COM1);
24.      return 1;
25.    }
26.    else
27.    {
28.      return 0;
29.    }
30.  }
31.
32.  //改成：
33.  uint32_t SerialKeyPressed(uint8_t *key)
34.  {
35.    if ( USART_GetFlagStatus(USART1, USART_FLAG_RXNE) != RESET)
36.    {
37.      *key = USART_ReceiveData(USART1);
38.      return 1;
39.    }
40.    else
41.    {
42.      return 0;
43.    }
44.  }
```

复制代码

在程序中我使用的升级方式是通过串口发送字符'1'(可自行修改)，串口收到字符'1'后，向指定的地址写入0xAAAA，然后使用软复位回到bootloader，bootloader中判断那个地址是不是0xAAAA，如果是则升级，升级完成后擦除这个地址的flash内容跳到APP，如果不是则跳直接回到APP执行，由于升级不可能频繁升，所以这里不用担心把falsh擦写坏。

AES256+PC软件：

完成ymodem协议传输代码后紧接着又有一个问题了，以往的代码中如果不需要用到升级功能我们会加上读保护，保证有一层明锁，不至于能直接读出代码，但是一旦有了APP和bootloader，虽然你代码还是能加上读保护，但是当APP变成一个bin文件给到客户升级的时候如果你不进行加密，很容易就给别人读出来，所以APP就要想办法再给到客户的时候又不至于那么容易让他破解，因此需要采用代码加密，AES是目前最流行的加密算法之一，破解也有一定的难度，关于AES的介绍看传送门
(http://baike.baidu.com/link?url=LHEswE_mMDL7EJv9-LeADTNvQvySI1H8ZaWKMZwBXJZnsNh9EbSM2xkbT8tbUgMQLbTQ-R9AjnS1I6ADK4f0EqonE5JKiB413tObuTd_nXm)
对代码加密有更加严格需求，想加暗锁的可以看下坛友写的一些方法

(http://www.openedv.com/thread-64685-1-1.html)
一开始也是想自己写AES来着，研究了网上的很多资料，偏理论性的东西实在太多了，后面找到了一个动图，可以很清晰的了解每个加密过程，看了后感觉茅塞顿开，动图贴附件中，需要自己写AES的可以根据这个动图来写。AES分加密和解密两个步奏，了解了原理以后偷懒上网查了一下有没现成的可以用，几次查找后发现了个好东西，如果自己写的话首先需要写解密到MCU上，然后PC需要自己写个软件来对bin进行加密，这份资料已经做完了这些，我只需要直接拿来用就好了，使用的是AES256，
想想也没那么简单，拿来就能直接用？作者提供MCU端AES算法倒是没问题，问题在PC端软件根本用不了，下载了他的软件后用VS编译器调试了下，发现作者不是纯用C#，界面用C#写的，AES的实现用C写的，用C代码生成dll给C#调用，问题就出现这个dll，不知道什么原因总之调用这个dll就报错，后面我重新将这部分C代码重新生成一个dll软件把平台改成X86后就可以正常使用了，接着只要将生成的hex文件导入这个软件，写入AES密匙后会自动生成一个加密后的代码，接着通过Ymodem传输给MCU，MCU边接收别解密，解密后写入flash，这个作者的原网址
(http://www.amobbs.com/thread-5069186-1-1.html)，他给的文件包无法直接用，我给的文件包估计也没法在你们的电脑上上直接用，所以只能用你们自己用电脑将C代码生成dll才能正常用了，怎么把C代码生成dll给C#调用，这个百度一大把，很容易(底下有教程，需要VS环境)。
下面是我加了AES的yemodem接收代码，由于是之前写的，现在只加了一些关键备注，附件中的源代码都有注释，这里截取关键的发出来

```
int32_t Ymodem_Receive (uint8_t *buf)
{                    //1024 + 5
        uint8_t bufferOut[16]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
  uint8_t packet_data[PACKET_1K_SIZE + PACKET_OVERHEAD], file_size[FILE_SIZE_LENGTH], *file_ptr, *buf_ptr;
  uint8_t *BufferIn;
        int32_t i, j, packet_length, session_done, file_done, packets_received, errors, session_begin, size = 0;
  uint32_t flashdestination, ramsource;

        aesDecInit();//AES解密初始化

  /* Initialize flashdestination variable */
  flashdestination = APPLICATION_ADDRESS; //APP代码的起始地址,APPLICATION_ADDRESS = 0x8005000,可在target界面根据情况设置
                                          //这些都在ymodem.h里面的宏进行设置


  for (session_done = 0, errors = 0, session_begin = 0; ;)//死循环直至文件数据包全部发送完成
  {
    for (packets_received = 0, file_done = 0, buf_ptr = buf; ;)
    {

      switch (Receive_Packet(packet_data, &packet_length, NAK_TIMEOUT))
      {
        case 0://成功接收到1K
          errors = 0;
          switch (packet_length)
          {
            /* Abort by sender */
            case - 1:  //接收失败
              Send_Byte(ACK);  //回复
              return 0;
            /* End of transmission */
            case 0:
              Send_Byte(ACK);//回复
              file_done = 1;
              break;
            /* Normal packet */
            default:   //接收成功
              if ((packet_data[PACKET_SEQNO_INDEX] & 0xff) != (packets_received & 0xff))
              {//序号00(文件名)
                Send_Byte(NAK);

              }
              else
              {
                if (packets_received == 0)//文件名(首包)
```

```
                {
                  /* Filename packet */
                  if (packet_data[PACKET_HEADER] != 0)//文件名字
                  {
                    /* Filename packet has valid data */
                    for (i = 0, file_ptr = packet_data + PACKET_HEADER; (*file_ptr != 0) && (i < FILE_NAME_LENGTH);)
                    {
                      FileName[i++] = *file_ptr++;//保存文件名
                    }
                    FileName[i++] = '\0';//字符串形式
                    for (i = 0, file_ptr ++; (*file_ptr != ' ') && (i < (FILE_SIZE_LENGTH - 1));)
                    {
                      file_size[i++] = *file_ptr++;//文件大小
                    }
                    file_size[i++] = '\0';
                    Str2Int(file_size, &size);//Convert a string to an integer

                    /* Test the size of the image to be sent */
                    /* Image size is greater than Flash size */
                    if (size > (USER_FLASH_SIZE + 1))
                    {
                      /* End session */
                      Send_Byte(CA);
                      Send_Byte(CA);
                      return -1;
                    }
                    /* erase user application area */
                                                                                FLASH_Unlock();                                         //解锁
                    FLASH_If_Erase(APPLICATION_ADDRESS);//This function does an erase of all user flash area
                                                                                FLASH_Lock();           //上锁

                                                                                Send_Byte(ACK);
                                                                                Send_Byte(CRC16);
                  }
                  /* Filename packet is empty, end session */
                  else
                  {
                    Send_Byte(ACK);
                    file_done = 1;
                    session_done = 1;
                    break;
                  }
                }
                /* Data packet */
                else  //文件信息保存完后开始接收数据
                {
                  memcpy(buf_ptr, packet_data + PACKET_HEADER, packet_length);

                                                                        /*------------------------------------------------------------------------*/
                                                                        BufferIn=buf;
                                                                        for (j = 0; j < packet_length; j += 16) //每次解密16字节
                                                                        {
                                                                                //解密数据包
                                                                                aesDecrypt(BufferIn,bufferOut); //由于参数使用的是指针,所以解密后依旧存在buf里面
                                                                                BufferIn+=16;
                                                                        }
                                                                        /*------------------------------------------------------------------------*/
```

```
103.
104.                   ramsource = (uint32_t)buf;
105.
106.                   /* Write received data in Flash */                                  //这个si
ze参数是自己加进入的,便于判断文件传输完成
107.                   if (FLASH_If_Write(&flashdestination, (uint32_t*) ramsource, (uint16_t) pack
et_length/4, size)  == 0)
108.                   {//写入FLASH
109.                     Send_Byte(ACK);
110.                   }
111.                   else /* An error occurred while writing to Flash memory */
112.                   {
113.                     /* End session */
114.                     Send_Byte(CA);
115.                     Send_Byte(CA);
116.                     return -2;
117.                   }
118.                 }
119.                 packets_received ++;
120.                 session_begin = 1;
121.               }
122.           }
123.           break;
124.         case 1:
125.           Send_Byte(CA);
126.           Send_Byte(CA);
127.           return -3;
128.         default://检验错误
129.           if (session_begin > 0)
130.           {
131.             errors ++;
132.           }
133.           if (errors > MAX_ERRORS)
134.           {
135.             Send_Byte(CA);
136.             Send_Byte(CA);
137.             return 0;
138.           }
139.           Send_Byte(CRC16); //发送校验值
140.           break;
141.       }
142.
143.       if (file_done != 0)
144.       {
145.         break;
146.       }
147.     }
148.     if (session_done != 0) //文件发送完成
149.     {
150.       break;
151.     }
152.   }
153.   return (int32_t)size;
154. }
```

复制代码

Hex合并以及Hex转bin

首次下载代码时为了方便需要合并bootloader和APP的hex文件，Hex合并本论坛有人已经提供了一个很好的方法(http://www.openedv.com/thread-70162-1-1.html),转成一个hex文件后通过上面那个PC软件，可以直接生成bin文件，之后通过超级终端接通串口传输过去就行了。

无线方式传输

我试过用电脑蓝牙传输(使用超级终端发送)，接收方也用蓝牙，可以实现升级文件的传输，但是一般客户不会使用电脑蓝牙来发送，只有笔记本有，台式需要蓝牙适配器，当然使用USB转串口来实现升级也是不错的选择，但

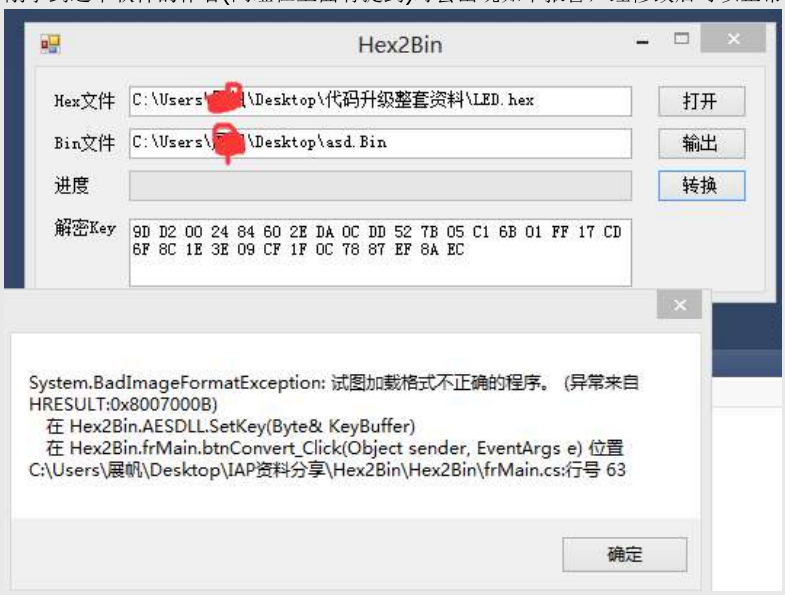
是需要个USB转串口芯片，也可以直接改修串口发送函数改成USB发送，加上USB驱动就行了。综合考虑还是在产品的APP通过ymodem协议发送会比较符合现在客户的主流需求，安卓系统的我在网上有找到java现成的ymodem发送代码，但是苹果系统如果使用OC进行开发，我目前还没有找到，希望有资源的坛友能提供，下面需要java代码的ymodem协议的可以在这个网址搜
(https://github.com/search?utf8=%E2%9C%93&q=ymodem)
安卓手机也可以下载有些通用的串口助手，有些也有带ymodem协议。至于其他的无线方式也是一样的。

整个升级过程分为以下步奏：

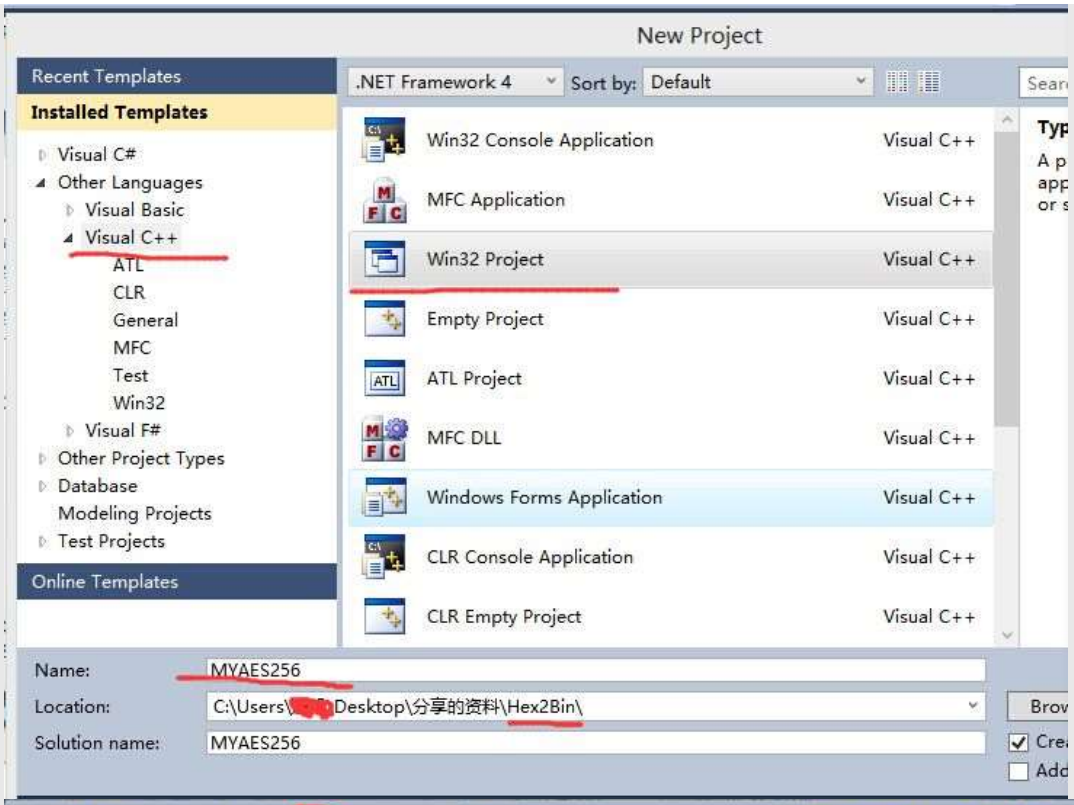
1.完成ymodem移植，修改官方ymodem串口发送以及修改flash的一些操作(操作前解锁，操作完上锁)，修改ymodem.h的宏配置，对应编译器的bootloader区和APP代码区的地址
2.移植aes.c的代码，添加到ymodem接收函数中
3.在target界面修改APP的代码区地址，写入跳回Bootloader的条件，接收到串口字符'1'后往指定地址写入0xAAAA，软复位在bootloader判断这个地址是否是0xAAAA(这个写入我是另外开辟了1Kflash专门用于存储掉电不丢失数据区，比如0xAAAA就是)，是则升级后跳回APP
4.修改附件提供的PC端AES256加密软件，用VS编译器重新将环境改为X86，C代码(.c文件)生成DLL供C#调用，C#代码中替换下关键字即可，如果我附件中的软件能直接用则不需要做任何修改(直接能用的概率不高)
5.合并两个hex文件后，通过PC软件加上16字节的密匙后自动生成bin
6.下载超级终端，设置为ymodem发送文件，首先发送两个字符'1'，进入bootloader后发送文件完成升级

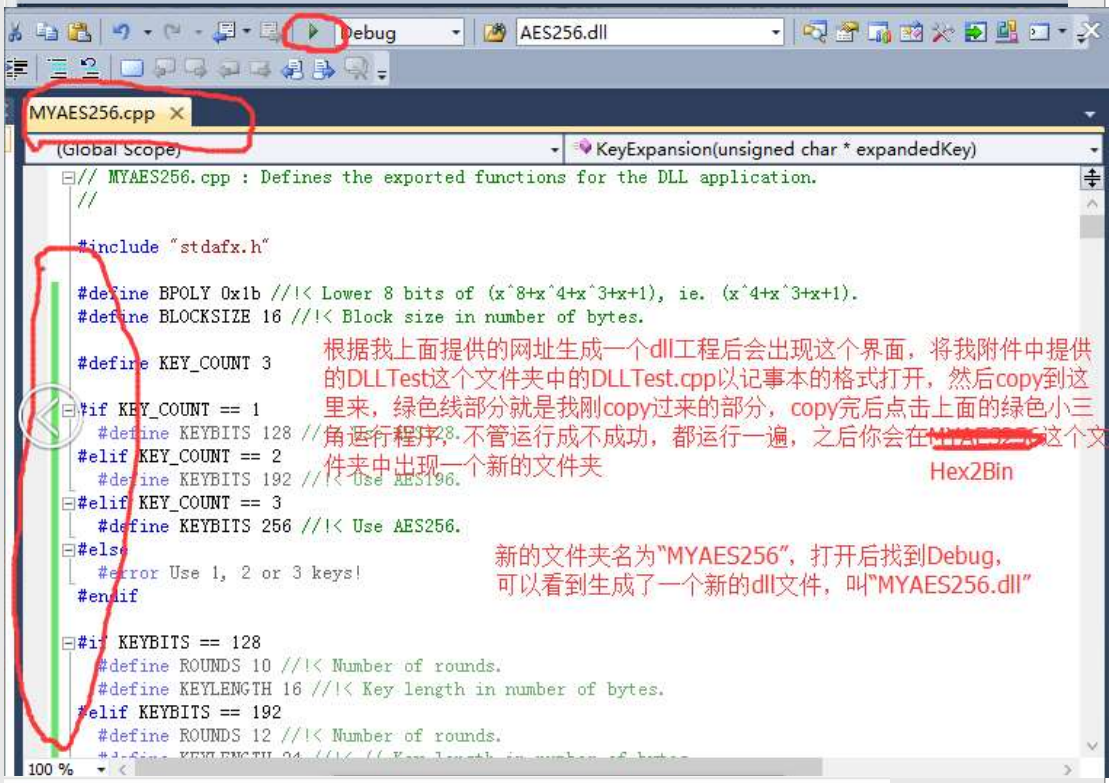
PC软件

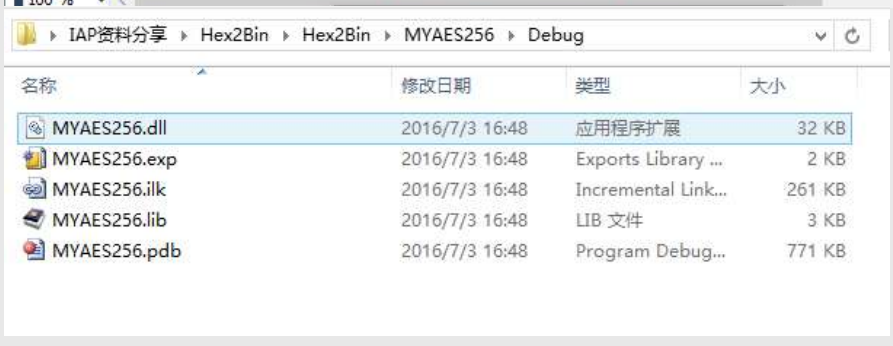
刚拿到这个软件的作者(网址在上面有提到)时会出现如下报警，经修改后可以正常使用。



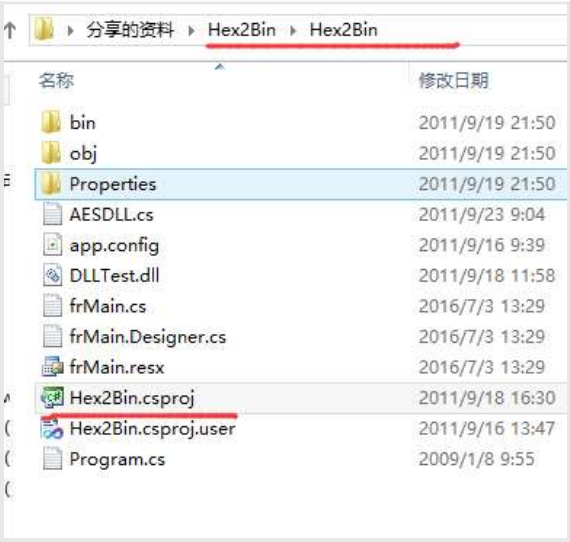
VS如何生成DLL文件让PC端软件能够正常使用，先根据下面这个网址操作一遍生成dll工程
http://jingyan.baidu.com/article/ff411625ad116612e48237a4.html


New Project
Recent Templates
Installed Templates
Visual C#
Other Languages
Visual Basic
Visual C++
ATL
CLR
General
MFC
Test
Win32
Visual F#
Other Project Types
Database
Modeling Projects
Test Projects
Online Templates
.NET Framework 4
Sort by: Default
Win32 Console Application
MFC Application
Win32 Project
Empty Project
ATL Project
MFC DLL
Windows Forms Application
CLR Console Application
CLR Empty Project
Visual C++
Name: MYAES256
Location: C:\Users\...\Desktop\分享的资料\Hex2Bin\
Solution name: MYAES256

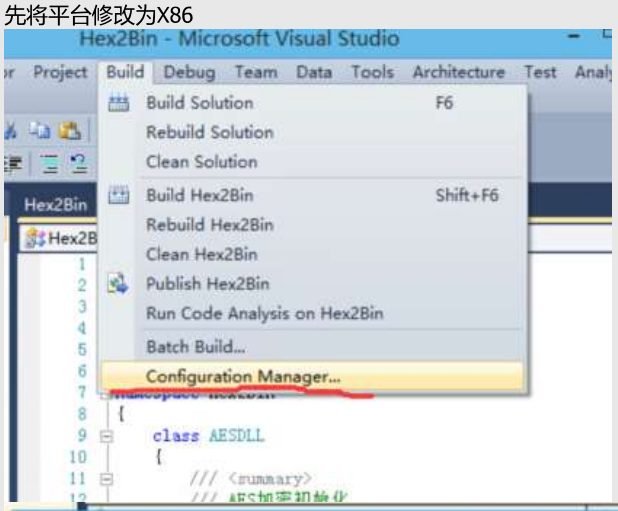

Debug
AES256.dll
MYAES256.cpp
(Global Scope)
KeyExpansion(unsigned char * expandedKey)
// MYAES256.cpp : Defines the exported functions for the DLL application.
#include "stdafx.h"
#define BPOLY 0x1b //!< Lower 8 bits of (x^8+x^4+x^3+x+1), ie. (x^4+x^3+x+1).
#define BLOCKSIZE 16 //!< Block size in number of bytes.
#define KEY_COUNT 3
#if KEY_COUNT == 1
#define KEYBITS 128 //!< Use AES128.
#elif KEY_COUNT == 2
#define KEYBITS 192 //!< Use AES196.
#elif KEY_COUNT == 3
#define KEYBITS 256 //!< Use AES256.
#else
#error Use 1, 2 or 3 keys!
#endif
#if KEYBITS == 128
#define ROUNDS 10 //!< Number of rounds.
#define KEYLENGTH 16 //!< Key length in number of bytes.
#elif KEYBITS == 192
#define ROUNDS 12 //!< Number of rounds.
100 %
根据我上面提供的网址生成一个dll工程后会出现这个界面，将我附件中提供的DLLTest这个文件夹中的DLLTest.cpp以记事本的格式打开，然后copy到这里来，绿色线部分就是我刚copy过来的部分，copy完后点击上面的绿色小三角运行程序，不管运行成不成功，都运行一遍，之后你会在[illegible]这个文件夹中出现一个新的文件夹
Hex2Bin
新的文件夹名为"MYAES256"，打开后找到Debug，可以看到生成了一个新的dll文件，叫"MYAES256.dll"

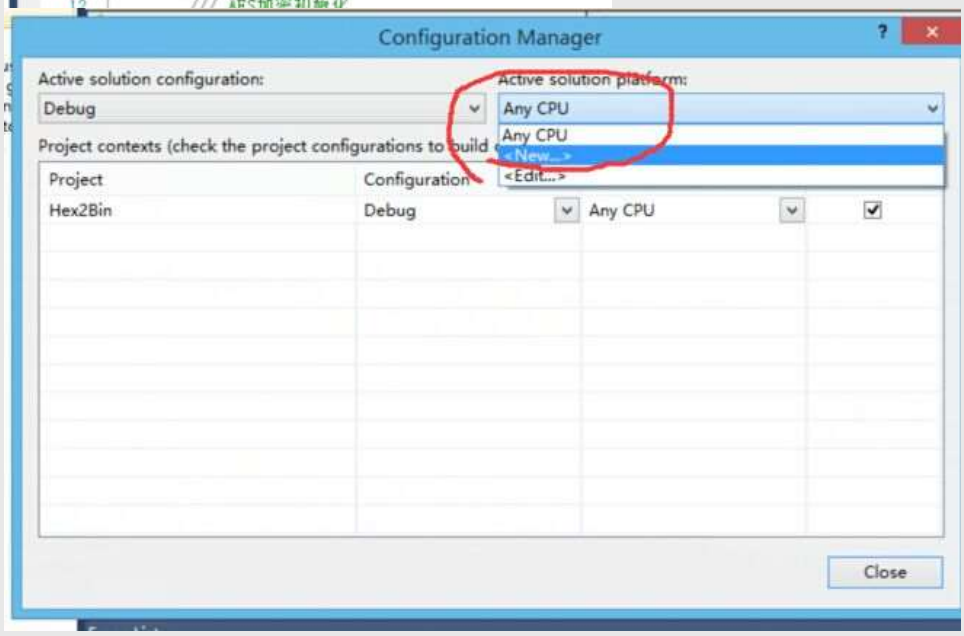

IAP资料分享 ▸ Hex2Bin ▸ Hex2Bin ▸ MYAES256 ▸ Debug
名称 修改日期 类型 大小
MYAES256.dll 2016/7/3 16:48 应用程序扩展 32 KB
MYAES256.exp 2016/7/3 16:48 Exports Library ... 2 KB
MYAES256.ilk 2016/7/3 16:48 Incremental Link... 261 KB
MYAES256.lib 2016/7/3 16:48 LIB 文件 3 KB
MYAES256.pdb 2016/7/3 16:48 Program Debug... 771 KB

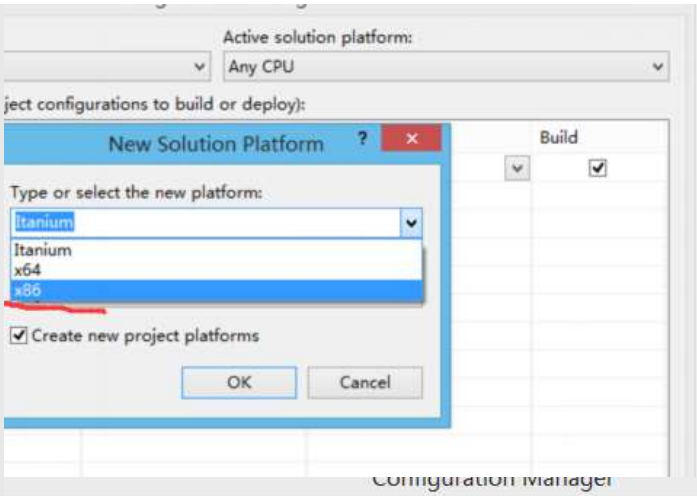
接着先打开项目工程

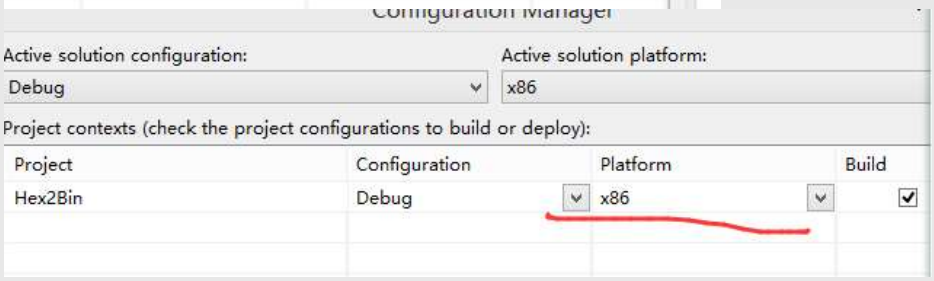

分享的资料 ▸ Hex2Bin ▸ Hex2Bin
名称
修改日期
bin 2011/9/19 21:50
obj 2011/9/19 21:50
Properties 2011/9/19 21:50
AESDLL.cs 2011/9/23 9:04
app.config 2011/9/16 9:39
DLLTest.dll 2011/9/18 11:58
frMain.cs 2016/7/3 13:29
frMain.Designer.cs 2016/7/3 13:29
frMain.resx 2016/7/3 13:29
Hex2Bin.csproj 2011/9/18 16:30
Hex2Bin.csproj.user 2011/9/16 13:47
Program.cs 2009/1/8 9:55

先将平台修改为X86

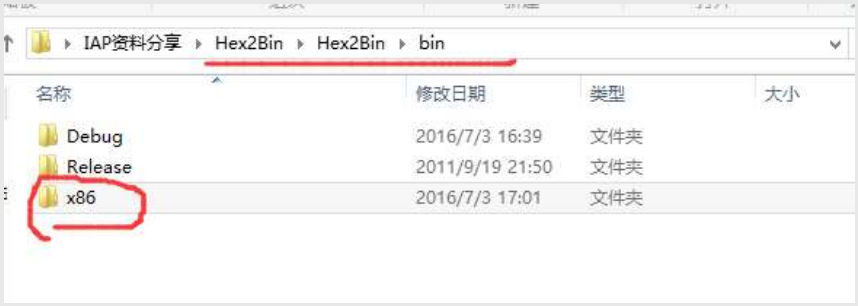

Hex2Bin - Microsoft Visual Studio
Project Build Debug Team Data Tools Architecture Test
Build Solution F6
Rebuild Solution
Clean Solution
Build Hex2Bin Shift+F6
Rebuild Hex2Bin
Clean Hex2Bin
Publish Hex2Bin
Run Code Analysis on Hex2Bin
Batch Build...
Configuration Manager...
class AESDLL


Configuration Manager
Active solution configuration:
Debug
Active solution platform:
Any CPU
Any CPU
<New...>
<Edit...>
Project contexts (check the project configurations to build
Project
Configuration
Hex2Bin
Debug
Any CPU
Close



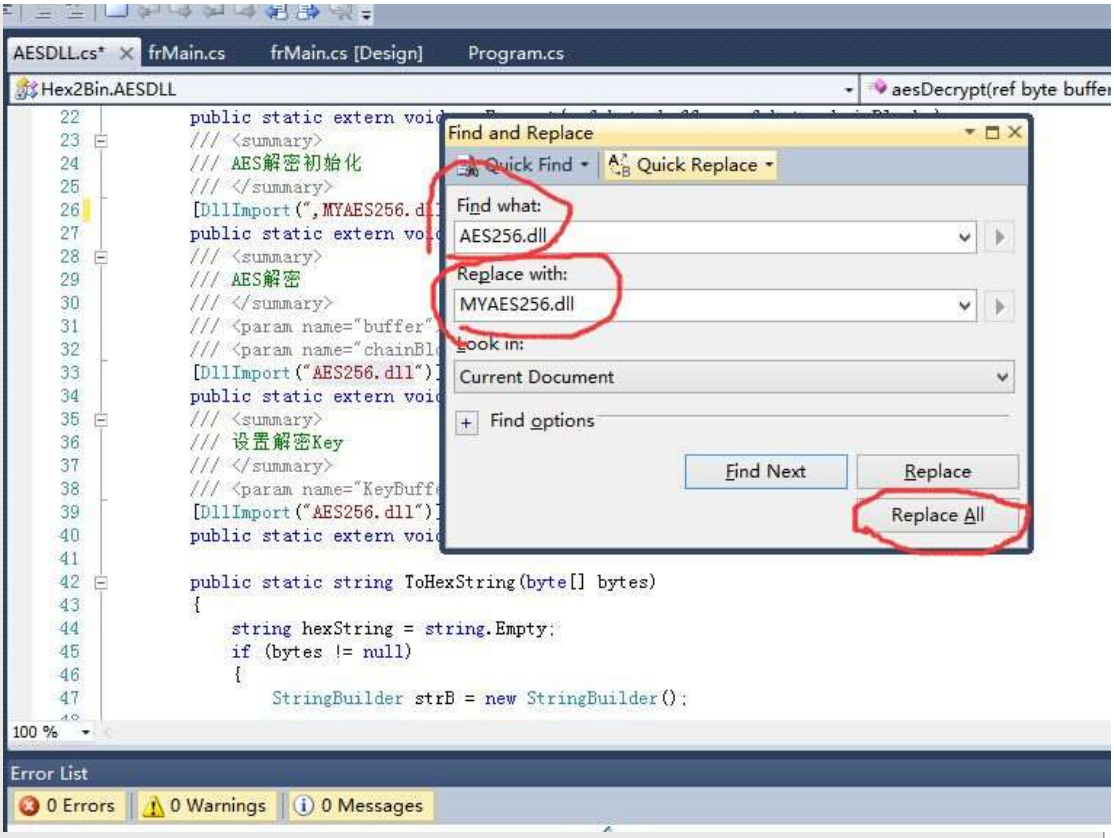


修改完后编译一遍，可以看到多了一个x86的文件夹

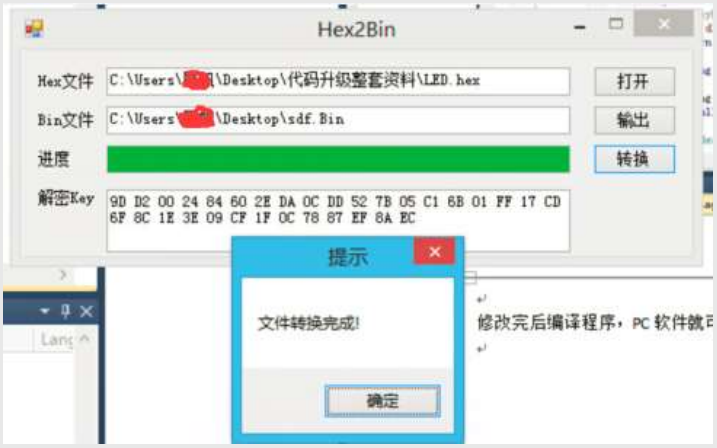


之后将Debug里面有关MYAES256的5个文件都copy到x86->debug里面，如下

接下来要做的就是用这个“MYAES256.dll”替代作者的那个“AES256.dll”了，替换关键字



修改完后编译程序，PC软件就可以正常运行了，密匙根据自己的MCU解密匙来写，如下



生成一个bin在桌面，之后用"超级终端"发送这个bin文件就可以了

```
================== Main Menu ============================

  Download Image To the STM32F0xx Internal Flash ------- 1

  Upload Image From the STM32F0xx Internal Flash ------- 2

  Execute The New Program ----------------------------- 3

============================================================

CCCCCCCC

 Programming Completed Successfully!
--------------------------------
 Name:LED.bin
 Size: 7872 Bytes
-------------------
```

升级资料分享.zip
15.23 MB, 下载次数: 105392