XRADIO Efuse Tool User Guide

Revision 2.2

Aug 08, 2020

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2017-11-23	Initial Version
2.0	2019-08-07	Add XR872 version
2.1	2019-10-11	Fix file name
2.2	2020-08-08	Format Optimize

Table 1- 1 Revision History

Contents

Tables

Figures

1 概述

1.1 EFUSE API 介绍

EFUSE 用于对 XR872 芯片的 eFuse 进行烧写操作，通过 UART 串口实现命令响应的收发。efuse_api.dll 封装了一些列操作的接口函数，在 efuse_api.h 头文件中可以查看对应的 API 介绍。

1.2 API 配置方法

下面以示例工程为例，简单介绍 efuse_api 的配置方法，总共有 efuse_api.dll、efuse_api.lib、efuse_api.h 三个文件需要配置。

1.2.1 efuse_api.dll

此文件是 API 的动态链接库，只需要将它与调用它的 exe 应用程序放在同一个目录下即可。

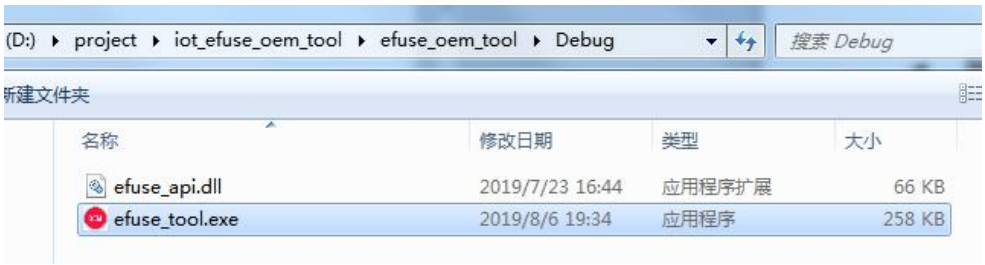

图 1-1 动态链接库 dll 文件

1.2.2 efuse_api.lib

此文件是 efuse_api.dll 动态链接库的导入库，并非静态链接库文件。需要将此文件放在工程目录下，并且在使用到 API 的文件中添加下面语句导入库文件。

```
#pragma comment(lib,"efuse_api.lib")
```

如图所示：

```
//

#pragma once

#include "afxwin.h"
#include "efuse_api.h"
#include "Comm.h"
#include "dlgEfuse709.h"
#include "dlgEfuse871.h"
#include "dlgEfuse872.h"
#include "afxcmn.h"
#pragma comment(lib,"efuse_api.lib")

const CString strVersion = (_T("Efuse tool version: 2.0.08061"));
const CString strCopyright = (_T("Copyright (C): XRadio Technology"));
```

图 1-2 添加导入库 lib 文件的代码修改

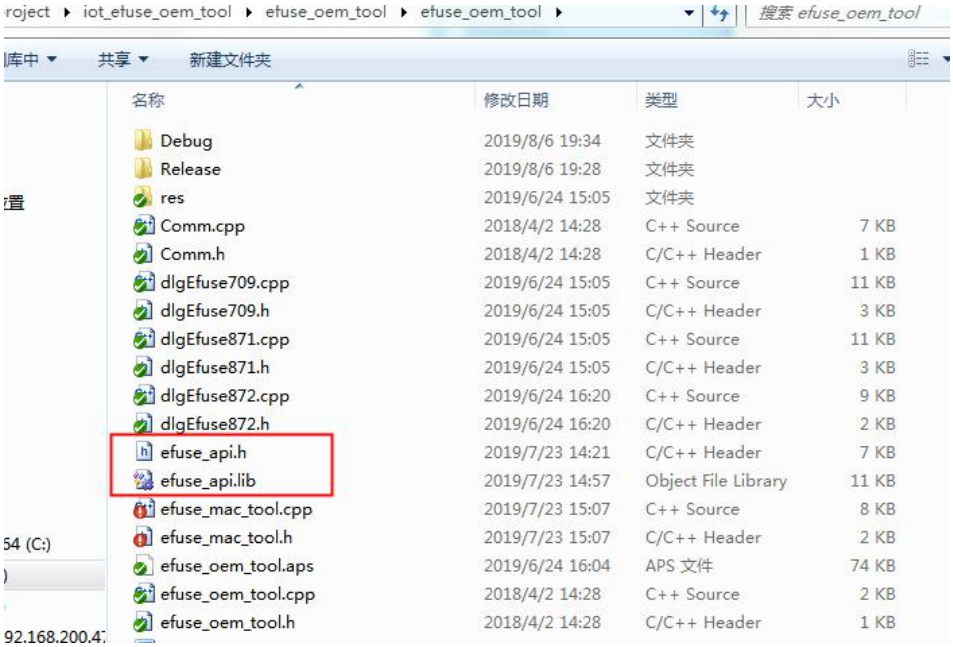

图 1-3 将 lib 文件放在工程中

1.2.3 efuse_api.h

此文件是 API 的头文件，包含对 API 的一些简单介绍。需要将此文件放在工程目录下，并在使用到 API 的文件中包含此文件即可。如图 1-2 和 1-3 所示。

2 API 介绍

2.1 EFUSE_GetMsg

原型：char* EFUSE_GetMsg()

描述：获取最后一次命令执行的状态信息。

输入参数：无

返回值：指向状态信息字符串的指针

示例：

```
m_strMsg.Format(_T("%s"), EFUSE_GetMsg());

GetDlgItem(IDC_STATUS)->SetWindowText(m_strMsg);
```

2.2 EFUSE_SetHashKey

原型：int EFUSE_SetHashKey(char* buf, int len)

描述：设置哈希密钥。

输入参数：buf - 哈希字符串的指针

len - 哈希字符串的长度，最大为 256

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
int ret = 0;

CString strKey = _T(“12345678”);

if (EFUSE_SetHashKey(m_strKey.GetBuffer(), m_strKey.GetLength()))

    ret = -1;

m_strKey.ReleaseBuffer();

Return ret;
```

2.3 EFUSE_InitEfuse

原型：int EFUSE_InitEfuse(int comNum, char* keyBuf, int len)

描述：初始化 EFUSE，包括串口和密钥数据，进入 EFUSE 模式。

输入参数：comNum - 串口号

keyBuf - 哈希字符串的指针

len - 哈希字符串的长度，最大为 256

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
int com, keyLen;

com = pMain->m_ComNum.GetCurSel();

keyLen = pMain->m_strKey.GetLength();

if (EFUSE_InitEfuse(com, pMain->m_strKey.GetBuffer(), keyLen))

{

    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());

    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);

    EFUSE_ExitEfuse();

    return -1;

}
```

2.4 EFUSE_ExitEfuse

原型：int EFUSE_ExitEfuse()

描述：退出 EFUSE 模式，关闭串口。

输入参数：无

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
int com, keyLen;

com = pMain->m_ComNum.GetCurSel();

keyLen = pMain->m_strKey.GetLength();

if (EFUSE_InitEfuse(com, pMain->m_strKey.GetBuffer(), keyLen))

{

    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());

    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);

    EFUSE_ExitEfuse();

    return -1;

}

pMain->m_strKey.ReleaseBuffer();
```

2.5 EFUSE_WriteHoscType

原型：int EFUSE_WriteHoscType(int hosc)

描述：写入 HOSC type。

输入参数：hosc：0 - 26M　1 - 40M　2 - 24M　3 - 52M　-1 - invalid

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
        if (EFUSE_WriteHoscType(pMain->m_iHoscVal))

        {

            pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());

            SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);

            EFUSE_ExitEfuse();

            return -1;

        }
```

2.6 EFUSE_ReadHoscType

原型：int EFUSE_ReadHoscType(int* hosc)

描述：读取 HOSC type。

输入参数：hosc：0 - 26M　1 - 40M　2 - 24M　3 - 52M　-1 - invalid

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
if (EFUSE_ReadHoscType(&(pMain->m_iHoscVal)))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
```

2.7 EFUSE_WriteScrBoot

原型：int EFUSE_WriteScrBoot(char* hash, int len)

描述：写入 secure boot 的哈希值。

输入参数：hash - 用来存储哈希字符串的数组，只有 32byte

len - 哈希字符串的长度，只能为 32

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
if (EFUSE_WriteScrBoot(pMain->m_strScrBootVal.GetBuffer(), 32))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
pMain->m_strScrBootVal.ReleaseBuffer();
```

2.8 EFUSE_ReadScrBoot

原型：int EFUSE_ReadScrBoot(char* hash, int len)

描述：读取 secure boot 的哈希值。

输入参数：hash - 存储哈希字符串的数组

len - 存储哈希字符串的数组长度，需要大于或等于 32

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
char buf[33];
if (EFUSE_ReadScrBoot(buf, sizeof(buf)))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
buf[32] = '\0';
pMain->m_strScrBootVal.Format(_T("%s"), buf);
```

2.9 EFUSE_WriteDcxoTrim

原型：int EFUSE_WriteDcxoTrim(char value)

描述：写入 DCXO TRIM 值。

输入参数：value - DCXO TRIM 值

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
BYTE dcxoTrim;
dcxoTrim = (BYTE)_tcstoul(pMain->m_strDcxoTrimVal, NULL, 16);
if (EFUSE_WriteDcxoTrim(dcxoTrim))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
```

2.10 EFUSE_ReadDcxoTrim

原型：int EFUSE_ReadDcxoTrim(char* value)

描述：读取 DCXO TRIM 值。

输入参数：value - 存放 DCXO TRIM 值的指针

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
char dcxoTrim;
if (EFUSE_ReadDcxoTrim(&dcxoTrim))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
pMain->m_strDcxoTrimVal.Format(_T("%02X"), dcxoTrim);
```

2.11 EFUSE_WritePoutCal

原型：int EFUSE_WritePoutCal(char* rfCal, int len)

描述：写入 POUT CAL 值。

输入参数：rfCal - POUT CAL 值的数组，只有 3byte，每 byte 仅低 7bit 有效

len - POUT CAL 值数组的长度，只能为 3

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
char poutCal[3];
poutCal[0] = (BYTE)_tcstoul(pMain->m_strPoutCal1, NULL, 16);
poutCal[1] = (BYTE)_tcstoul(pMain->m_strPoutCal2, NULL, 16);
poutCal[2] = (BYTE)_tcstoul(pMain->m_strPoutCal3, NULL, 16);
if (EFUSE_WritePoutCal(poutCal, 3))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
```

2.12 EFUSE_ReadPoutCal

原型：int EFUSE_ReadPoutCal(char* rfCal, int len)

描述：读取 POUT CAL 值。

输入参数：rfCal - 存放 POUT CAL 值的数组

len - 存放 POUT CAL 值数组的长度，大于或等于 3

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
char poutCal[3];
if (EFUSE_ReadPoutCal(poutCal, sizeof(poutCal)))
{
    pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
    SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
    EFUSE_ExitEfuse();
    return -1;
}
pMain->m_strPoutCal1.Format(_T("%02X"), poutCal[0]);
```

2.13 EFUSE_WriteMacAddr

原型：int EFUSE_WriteMacAddr(char* mac, int len)

描述：写入 MAC 地址。

输入参数：mac - MAC 地址数组的指针，只有 6byte

len - MAC 地址数组的长度，只能为 6

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
        char mac[6];
        int i;
        CString strLeft, strRight;
        strRight = pMain->m_strMacAddr;
        for (i = 0;i < 6;i++)
        {
            strLeft = strRight.Left(2);
            strRight = strRight.Right(strRight.GetLength() - 2);
            mac[i] = (BYTE)_tcstoul(strLeft, NULL, 16);
        }
        if (EFUSE_WriteMacAddr(mac, 6))
        {
            pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
            SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
            EFUSE_ExitEfuse();
            return -1;
        }
```

2.14 EFUSE_ReadMacAddr

原型：int EFUSE_ReadMacAddr(char* mac, int len)

描述：读取 MAC 地址。

输入参数：mac - 存储 MAC 地址的数组

len - 存储 MAC 地址数组的长度，大于或等于 6

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
        int i;
        char mac[6];
        CString strTmp;
        pMain->m_strMacAddr = _T("");
        if (EFUSE_ReadMacAddr(mac, sizeof(mac)))
        {
            pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
            SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
            EFUSE_ExitEfuse();
            return -1;
        }
        for (i = 0;i < 6;i++)
        {
            strTmp.Format(_T("%02X"), mac[i]);
            pMain->m_strMacAddr += strTmp;
        }
```

2.15 EFUSE_WriteUserArea

原型：int EFUSE_WriteUserArea(char* data, int startAddr, int len)

描述：从指定的位地址开始写入用户区数据。

输入参数：data - 存储用户区数据的数组

startAddr - 起始的位地址，0~600，单位为 bit

len - 需要写入的用户区数据长度，1~601，单位为 bit

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
    int i;
    BYTE uaData[76];
    CString strLeft, strRight;
    WORD startBitAddr, bitLen, byteLen;
    CString strUserAreaData, strStartBitAddr, strBitLen;
    strUserAreaData = pMain->m_strUserAreaData0;
    strStartBitAddr = pMain->m_strStartBitAddr0;
    strBitLen = pMain->m_strBitLen0;
    strRight = strUserAreaData;
    startBitAddr = (WORD)_tcstoul(strStartBitAddr, NULL, 0);
    bitLen = (WORD)_tcstoul(strBitLen, NULL, 0);
    byteLen = bitLen / 8;
    if (bitLen % 8)  byteLen++;
    for (i = 0;i < byteLen;i++)
    {
        strLeft = strRight.Left(2);
        strRight = strRight.Right(strRight.GetLength() - 2);
        uaData[i] = (BYTE)_tcstoul(strLeft, NULL, 16);
    }
    if (EFUSE_WriteUserArea((char*)uaData, startBitAddr, bitLen))
    {
        pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
        SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
        EFUSE_ExitEfuse();
        return -1;
    }
    return 0;
```

2.16 EFUSE_ReadUserArea

原型：int EFUSE_ReadUserArea(char* data, int startAddr, int len)

描述：从指定的位地址开始读取用户区数据。

输入参数：data - 存储用户区数据的数组

startAddr - 起始的位地址，0~600，单位为 bit

len - 需要写入的用户区数据长度，1~601，单位为 bit

返回值：状态值，0 - 成功，非 0 - 失败

示例：

```
    int i;
    BYTE uaData[76];
    CString strTmp;
    WORD startBitAddr, bitLen, byteLen;
    CString strUserAreaData, strStartBitAddr, strBitLen;
    strUserAreaData = pMain->m_strUserAreaData0;
    strStartBitAddr = pMain->m_strStartBitAddr0;
    strBitLen = pMain->m_strBitLen0;
    strUserAreaData = _T("");
    startBitAddr = (WORD)_tcstoul(strStartBitAddr, NULL, 0);
    bitLen = (WORD)_tcstoul(strBitLen, NULL, 0);
    byteLen = bitLen / 8;
    if (bitLen % 8)  byteLen++;
    if (EFUSE_ReadUserArea((char*)uaData, startBitAddr, bitLen))
    {
        pMain->m_strMsg.Format(_T("%s"), EFUSE_GetMsg());
        SendMessage(pMain->m_hWnd, EFUSE_MSG, NULL, NULL);
        EFUSE_ExitEfuse();
        return -1;
    }
    for (i = 0;i < byteLen;i++)
    {
        strTmp.Format(_T("%02X"), uaData[i]);
        strUserAreaData += strTmp;
    }
    return 0;
```

2.17 EFUSE_WriteUserArea872

原型：int EFUSE_WriteUserArea872(char* data, int startAddr, int len)

描述：从指定的位地址开始写入用户区数据。

输入参数：data - 存储用户区数据的数组

startAddr - 起始的位地址，0~351，单位为 bit

len - 需要写入的用户区数据长度，1~352，单位为 bit

返回值：状态值，0 - 成功，非 0 - 失败

示例：可参考 EFUSE_WriteUserArea()接口的使用示例

2.18 EFUSE_ReadUserArea872

原型：int EFUSE_ReadUserArea872(char* data, int startAddr, int len)

描述：从指定的位地址开始读取用户区数据。

输入参数：data - 存储用户区数据的数组

startAddr - 起始的位地址，0~351，单位为 bit

len - 需要写入的用户区数据长度，1~352，单位为 bit

返回值：状态值，0 - 成功，非 0 - 失败

示例：可参考 EFUSE_ReadUserArea()接口的使用示例

3 示例工具

考虑到简洁方便，示例工程使用 MFC 编写，编译环境为 VS2008。

示例工具界面如下：

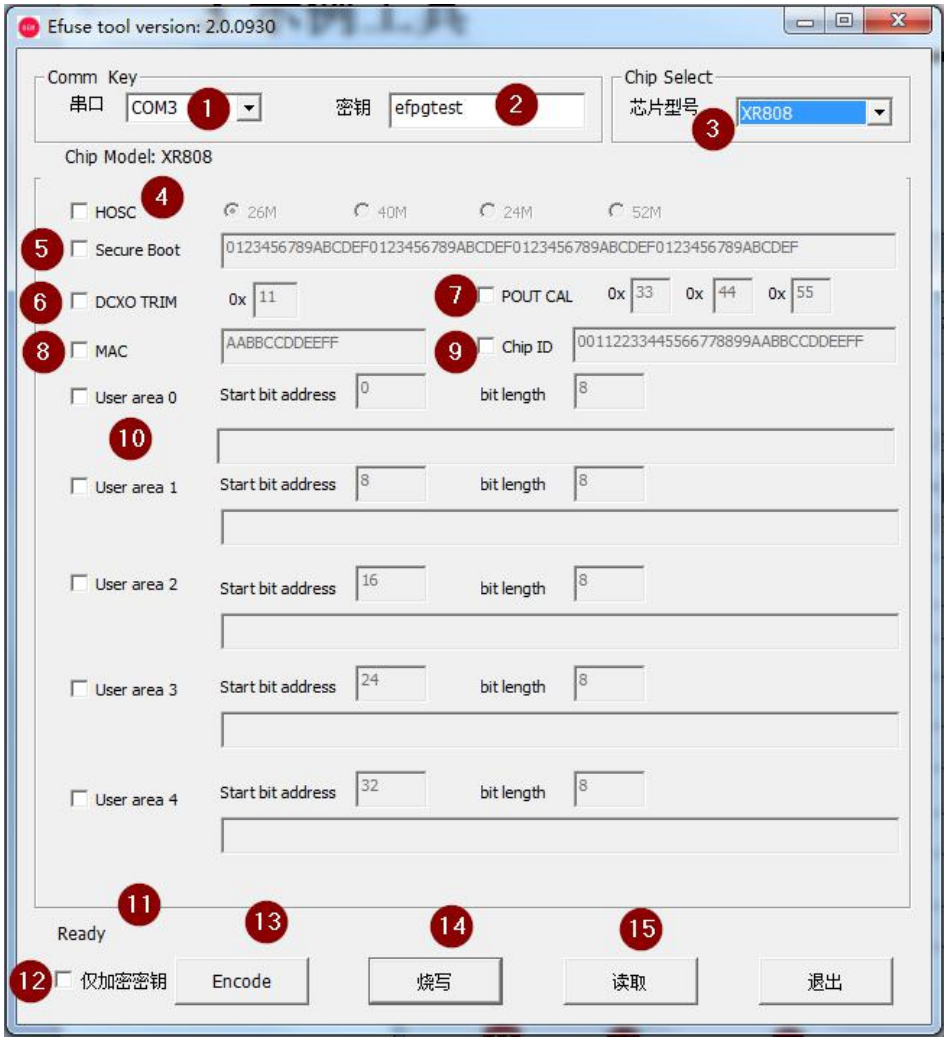

图 3-1 示例工具截图

1.串口选择；

2.密钥字符串；

3.选择芯片；

4.选择是否读写 HOSC;

5.选择是否读写 Secure Boot；

6.选择是否读写 DCXO TRIM；

7.选择是否读写 POUT CAL；

8.选择是否读写 MAC 地址；

9.选择是否读取 Chip ID；

10.选择是否读写 User Area，可以同时选择最多 5 个地址进行读写；

11.信息栏；

12.加密测试选择，勾选上时，点击“Encode”按钮，会对密钥进行单独加密，结果显示在信息栏。没有勾选上时，点击“Encode”按钮，会在信息栏显示上一次命令的加密结果；

13.Encode 加密测试按钮；

14.启动烧写功能按钮；

15.启动读取烧写结果功能按钮；