

© 2013 RealTEK Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of RealTEK Semiconductor Corp.

Mass Production Kit

EFUSE APIs Reference

Document Number: 01-001-M4-0005

Naming Rule: Reference Internal Engineer Development Document Code

Example: 01 0001 H1 01

Serial Number

High Level Design Firmware

RTL819X

Wireless NIC

Document Code Management

			Code
RTL Document		Functional Spec	F1
		High Level Design	H1
		Low Level Design	L1
		Test Plan, Test Report	T1
		Guideline	G1
		Maintain	M1
		Decisions , Meeting Notes	D1

Realtek Semiconductor Corp.

Headquarters

1F, No. 2, Industry East Road IX, Science-based

Industrial Park, Hsinchu, 300, Taiwan, R.O.C.

Tel: 886-3-5780211 Fax: 886-3-5776047

<http://www.realtek.com.tw/>

Change History

Version	Editor	Date	Remarks
0.001	Vincent Lan	2013/01/25	EFUSE READ/WRITE API FAQ
0.002	Vincent Lan and Kordan Ou	2013/02/07	Revised
0.003	Kordan	2013/04/03	APIs for RTL8761

REALTEK Confidential

Table of Contents:

Document Number: 01-001-M4-0005	2
Change History	3
Table of Contents:	4
1. Basic API Introduction	5
1.1. Build Environment	5
1.2. Device Under Test	5
1.3. Essential APIs	5
2. Essential APIs to initialize the DUT	6
3. EFUSE Operation Functions	8
4. A Better EFUSE Read/Write Method	9
4.1. Read/Write Whole Efuse	9
4.2. Read/Write One Byte of Efuse	10
4.3. WriteEfuse/ReadEfuse are Deprecated	10
5. A Sample Application for EFUSE Read/Write	11
5.1. BYTE Mode	11
5.2. EFUSE Mode	12
5.3. FILE Mode	13
5.4. UPDATE (important)	14

1. Basic API Introduction

1.1. Build Environment

Microsoft® Visual Studio® 6.0 is recommended for fully compatible with Microsoft Foundation Class (MFC) Library. Newer versions of Visual Studio® need to import Microsoft Foundation Class (MFC) Library. Visual Studio® 2008, 2012 test build ok, no need to install packages other than MFC but project setting configuration is required.

Visual Studio® Project Setting (based on Visual Studio® 2008):

- Enable C++ Exceptions : No
- Security Check : No
- Character Set : Use Multi-Byte
- Enable Function Level Linking : No
- Enable Incremental Linking : No
- Additional Dependencies : msvcr.lib setupapi.lib mp819xbd.lib cfgmgr32.lib

1.2. Device Under Test

RTL8188EU is used as the DUT in this document.

1.3. Essential APIs

Table 1. Essential API Name Table

Direction	API Name	Remark
General		
	StartDriverService	
	StartDUT	
	OpenDUT	
	CloseDUT	
	StartTest	
	StopTest	
	QueryChipIDandVer	
	DLL_CheckAutloadStatus	
	DLL_ReadEFuseUtilization	
EFUSE		
	ReadEFuse	
	ReadEFuseByte	
	WriteEFuse	
	WriteEFuseByte	
	UpdateEFuse	
	OpenBTSoloDUT	RTL8761 Only
	CloseBTSoloDUT	RTL8761 Only
	SelectBTSoloMap	RTL8761 Only

2. Essential APIs to initialize the DUT

2.1. WLAN EFUSE

- **BOOL StartDriverService()**

This function attempts to start the service of the protocol driver mp8185s.

- **<Note>**

1. It MUST be called first before using other API exported in this DLL.

- **<Example>**

```
- #define MP_QUIT() do { PostMessage(WM_QUIT); return FALSE; } while(0)
- if(!StartDriverService())
- {
-     MP_QUIT();
- }
```

- **PVOID StartDUT(UINT32 ChipID,
 UINT32 ChipVersion)**

This function opens the connection with specific DUT and get the handle of the DUT. It will return a NULL pointer if specified **ChipID** and **ChipVersion** is not found, otherwise, it will allocate memory for a **ADAPTER** object and return a pointer to this object. Other function who needs the argument, LPADAPTER AdapterObject, shall use the pointer returned by **StartDUT()**.

- **<Note>**

1. It should be invoked before other API needs the input argument, LPADAPTER AdapterObject.

2. Open connection with specific DUT with specified **ChipID** and **ChipVersion**

- **<Example>**

```
- // For RTL8188EU
- UINT32 ChipID = 0x8188
- UINT32 ChipVersion = 0x2 //PCIE: 0x1, USB: 0x2, SDIO: 0x3
- if(StartDUT(ChipID, ChipVersion) == NULL)
- {
-     MP_QUIT();
- }
```

- **int CloseDUT(LPADAPTER AdapterObject)**

This function closes the connection with DUT and release the handle and memory block allocated in **StartDUT()** for the DUT. It will return TRUE/FALSE to show operation success or fail.

- **<Note>**

1. It MUST be invoked when we don't need to access the AdapterObject any more before terminating the calling program. Otherwise, a memory leak problem will be encountered.

- **<Example>**

```
- //
- // wchar wszAdapterName = L"\\Device\\mp8185s_{08E0D84D-1B31-4D32-93B2-0AE05C90192D}";
- //
- LPADAPTER pAdapter = OpenDUT(wszAdapterName);
- StartTest(pAdapter);
- StopTest(pAdapter);
- CloseDUT(pAdapter);
```

- **int StartTest(LPADAPTER AdaterObject)**
This function initializes the DUT to the MP test mode. It will return TRUE/FALSE to show operation success or fail.
- **<Note>**
1.We SHOULD invoked this function before other Mass Production related test item, for example, Packet Tx test, Packet Rx test, Continuous Tx test, Single Carrier Tx test, Carrier Suppression Tx test and so on .
- **int StopTest(LPADAPTER AdapterObject)**
This function change the DUT from the test mode into normal mode. It will return TRUE/FALSE to show operation success or fail.
- **<Note>**
1.This function MUST be invoked if StartTest() is had been called before.
- **int QueryChipIDandVer(LPADAPTER AdapterObject, ULONG* ChipID, ULONG* ChipVer)**
This function queries the Chip ID and Version from the driver. It will return TRUE/FALSE to show operation success or fail.
- **int DLL_CheckAutloadStatus(LPADAPTER AdapterObject, unsigned char* AutoloadFail)**
This routine check autload status fail or success. It will return TRUE/FALSE to show operation success or fail.
1 : Autoload Fail
0 : Autoload Success.
- **int DLL_ReadEFuseUtilization(LPADAPTER AdapterObject, unsigned int* EFuseUtilize)**
This routine get the EFUSE utilization, bit 0~15 for how many bytes used, bit 16~31 for percentage. It will return TRUE/FALSE to show operation success or fail.

2.2. BT EFUSE (RTL8761)

<Example>

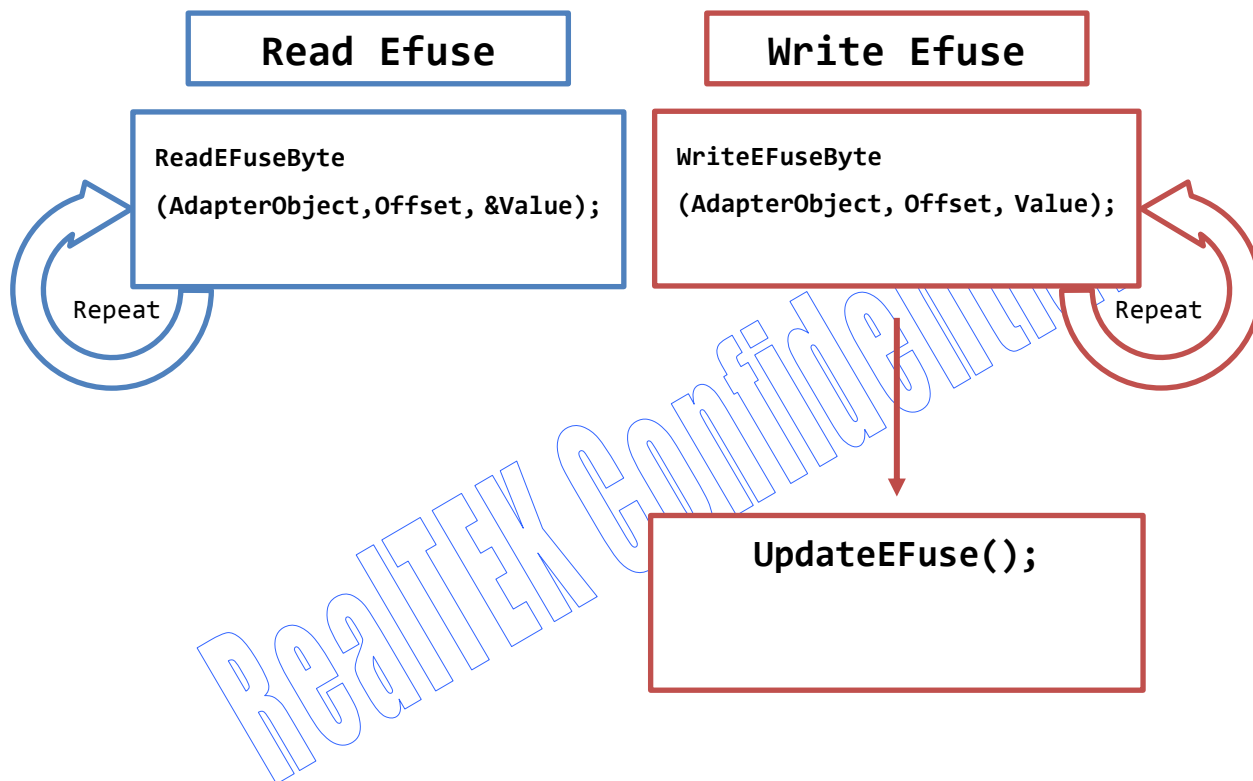
```
status1 = OpenBTSoloDUT();
// System Map
result1 = SelectBTSoloMap(0);
// The first argument is NULL because WLAN adapter is not necessary
status 2 = WriteBTEFuseByte(NULL, 0x10, 0x99);
status 3 = UpdateBTEfuse(NULL);
// BT Map
result2 = SelectBTSoloMap(1);
status 4 = WriteBTEFuseByte(NULL, 0x11, 0x88);
status 5 = ReadBTEFuseByte(NULL, 0x11, &returnedValue);
status 6 = UpdateBTEfuse(NULL);
status 7 = CloseBTSoloDUT();
```

3. EFUSE Operation Functions

- **int RTLMP_API ReadEFuse(LPADAPTER AdapterObject, char* EFuse)**
 (Deprecated) This function support to read EFUSE shadow memory from NIC side. You can not read real Efuse content. Driver will assist to protect you to execute incorrect operation on EFUSE area.
- **int RTLMP_API ReadEFuseByte(LPADAPTER AdapterObject, UCHAR Offset, UCHAR* Value)**
 This function reads the value of the byte offset specified from EFUSE shadow memory. It will return TRUE/FALSE to show operation success or fail.
- **int RTLMP_API WriteEFuse(LPADAPTER AdapterObject, UCHAR *EFuse)**
 (Deprecated) This function reads allow user to send specific number of bytes Efuse table to driver. At the time, driver only save the content in shadow memory.
- **int RTLMP_API WriteEFuseByte(LPADAPTER AdapterObject, UCHAR offset, UCHAR Value)**
 This function writes 1-byte value into the byte offset of EFUSE shadow memory. It will return TRUE/FALSE to show operation success or fail.
- **int RTLMP_API UpdateEFuse (LPADAPTER AdapterObject)**
 This function allow user to update previous content in shadow memory to real EFUSE area. Driver will automatically compare shadow map and logical map. And we **only write different area into efuse to prevent from wasting EFuse space.**
- **int RTLMP_API OpenBTSoloDUT()**
 When programming a **BT solo card** (e.g., RTL8761), this API loads BT EFUSE DLL first and then open its corresponding port. The returned value is a Boolean value actually.
- **int RTLMP_API CloseBTSoloDUT()**
 The API called at the end of EFUSE operation of a **BT solo card** (e.g., RTL8761). The returned value is a Boolean value actually.
- **int RTLMP_API SelectBTSoloMap(ULONG Map)**
 RTL8761 has two BT map. **It should be called explicitly before WriteEFuseByteBT().** The successful returned value is the map number being used, otherwise "-1" returned.
 Map #0: **System Map**
 Map #1: **BT Map**

4. A Better EFUSE Read/Write Method

4.1. Read/Write Whole Efuse



4.2. Read/Write One Byte of Efuse

Read One Byte

```
ReadEFuseByte(LPADAPTER AdapterObject,  
              UCHAR      Offset,  
              UCHAR*     Value)
```

Write One Byte

```
WriteEFuseByte(LPADAPTER AdapterObject,  
              UCHAR      Offset,  
              UCHAR      Value)
```

↓

```
UpdateEFuse();
```

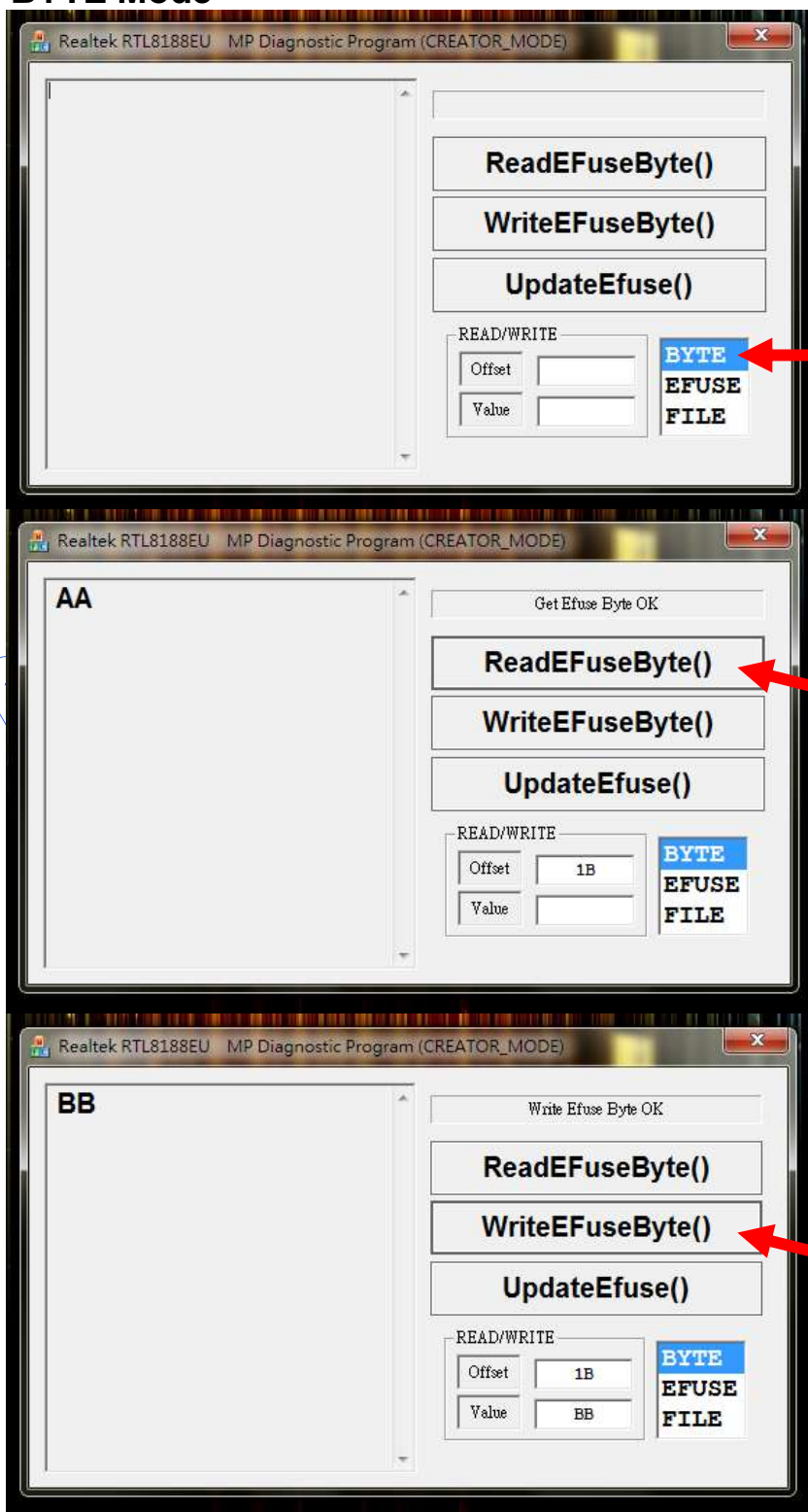
4.3. WriteEfuse/ReadEfuse are Deprecated

Because the arguments of WriteEfuse/ReadEfuse don't include the Efuse length, there is a vulnerability that the invalid memory might be accessed and cause UI crash.

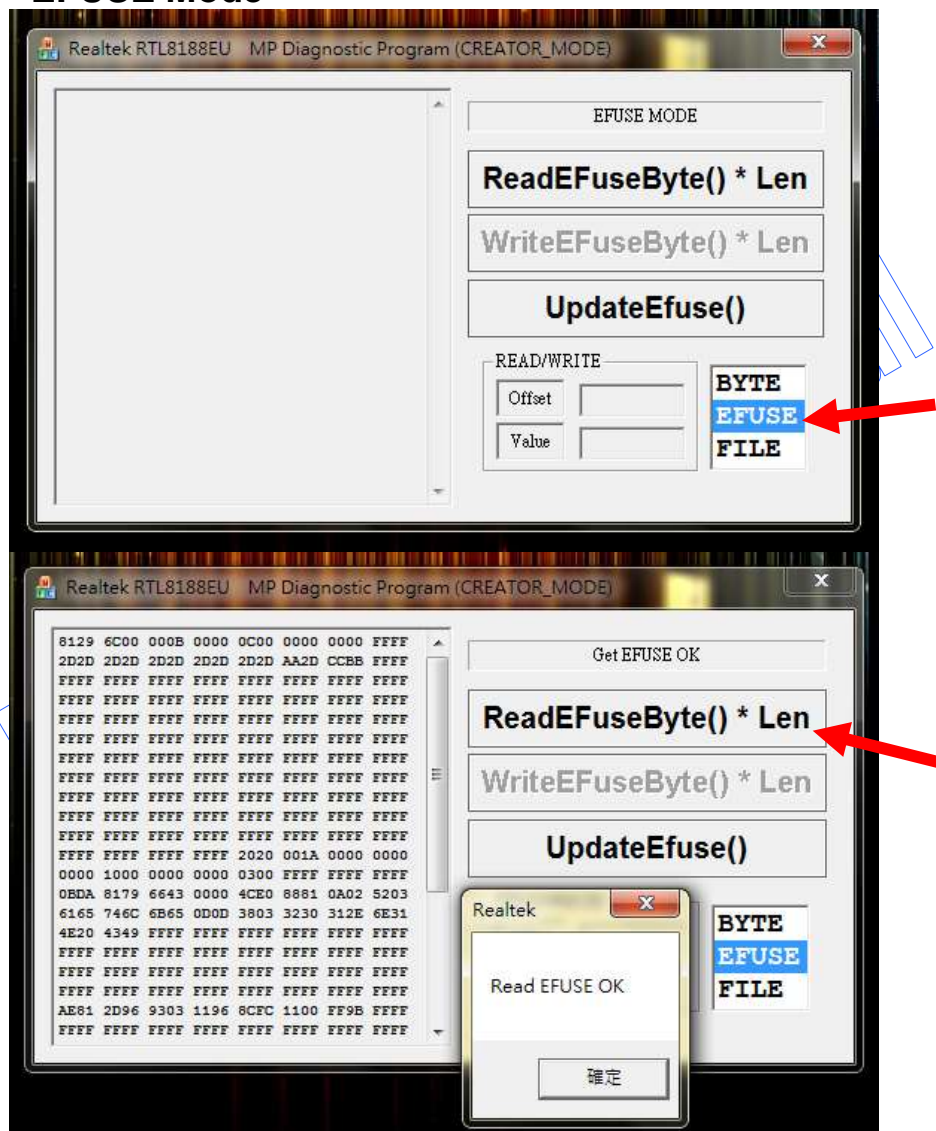
Either for reading/writing a full map or one byte, we could implement the methods by repeatedly calling WriteEfuseByte/ReadEfuseByte APIs instead. We recommend using WriteEfuseByte/ReadEfuseByte directly for better flexibility.

5. A Sample Application for EFUSE Read/Write

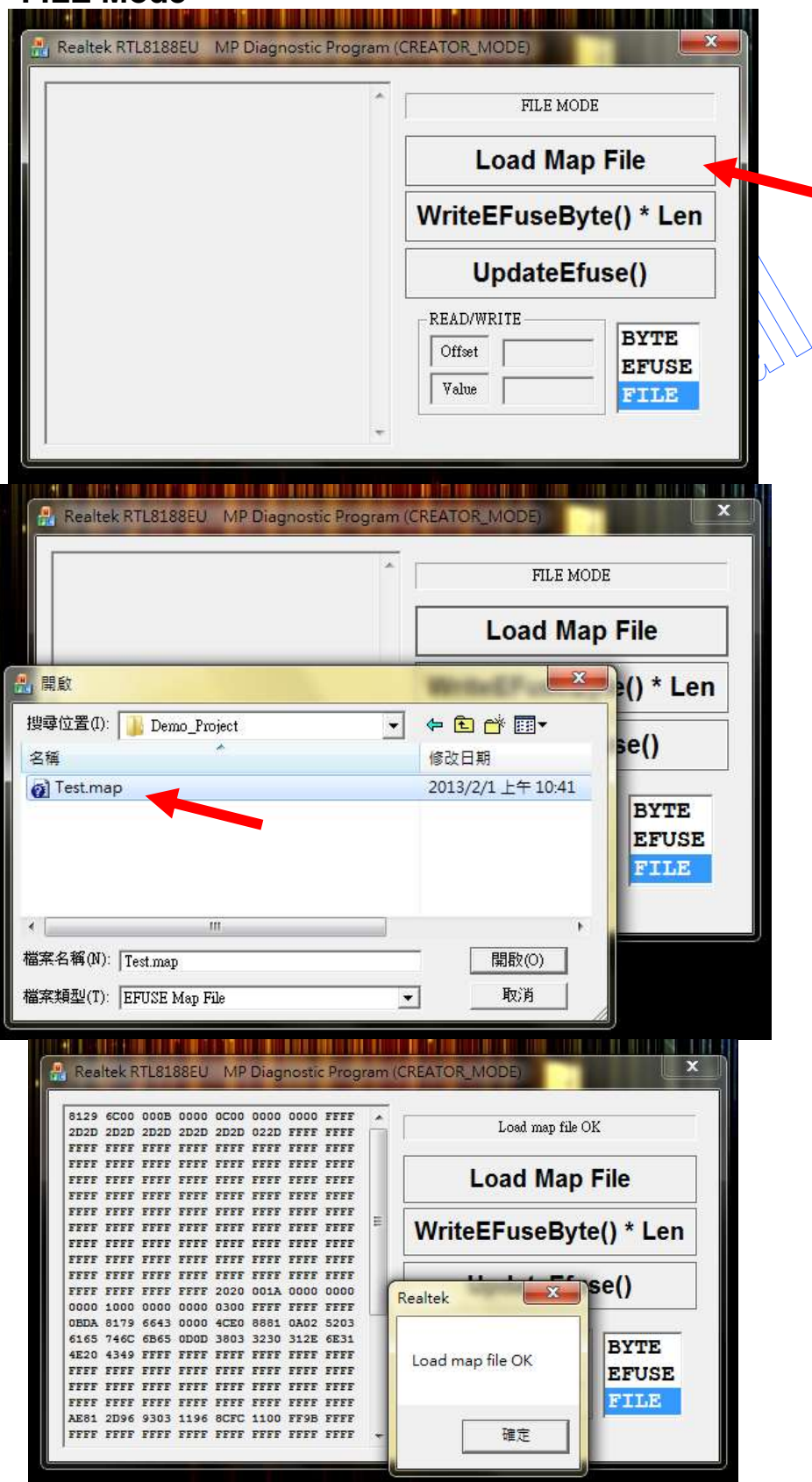
5.1. BYTE Mode



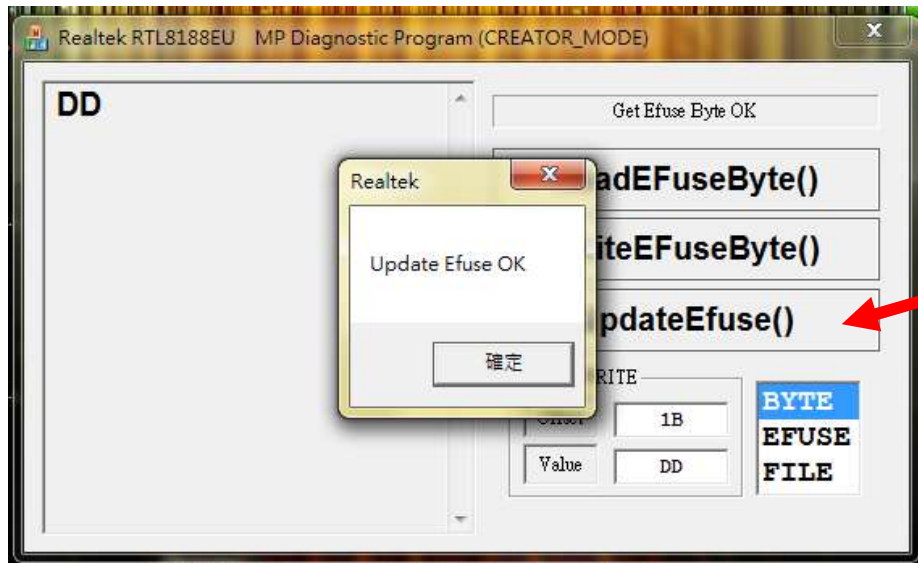
5.2. EFUSE Mode



5.3. FILE Mode



5.4. UPDATE (important)



All changes are not permanently programmed until API `UpdateEfuse()` is called. Otherwise changes are lost after re-plugging the NIC.