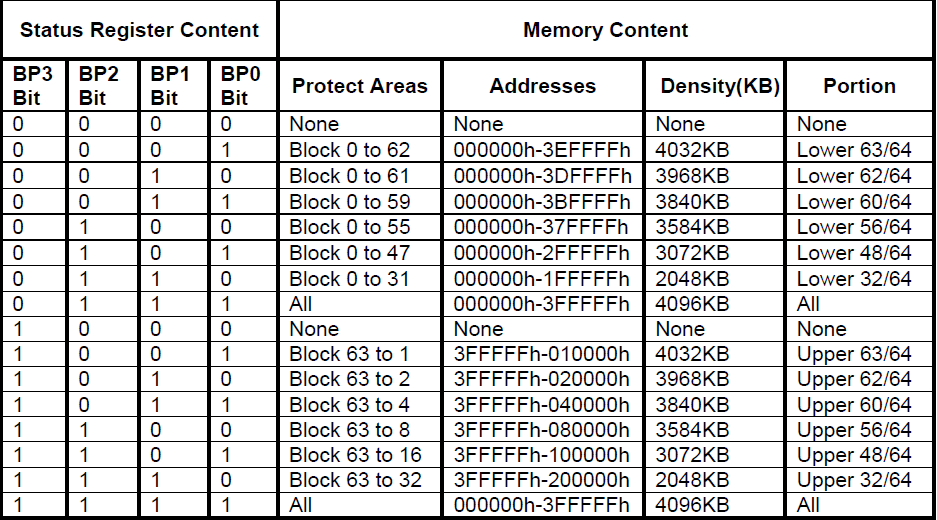
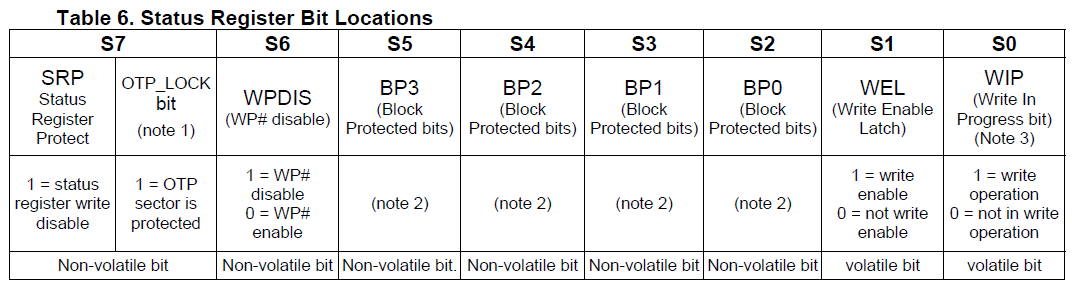
1. **先简单介绍下FLASH存储原理。**

由于写FLASH只能由1到0，然而0到1的动作是由擦除完成。

如果FLASH某个地址的数据是0X2F，那么写入0XF2，结果FLASH此处的数据是0X22。





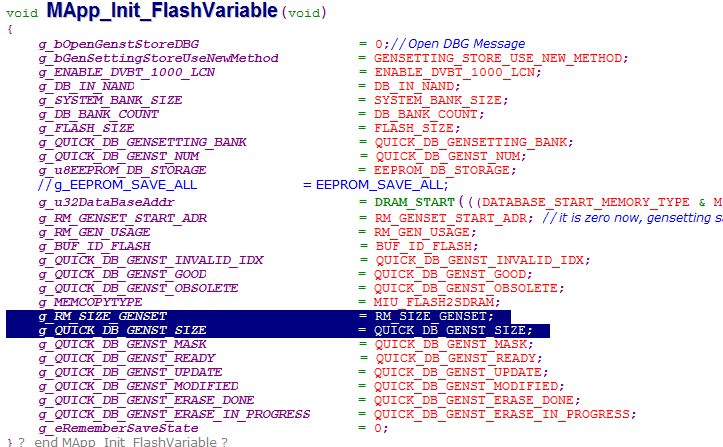
1. **Gensetting数据结构。**

注意：A.QUICK\_DB\_GENST\_SIZE\*QUICK\_DB\_GENST\_NUM大小为2个BANK。

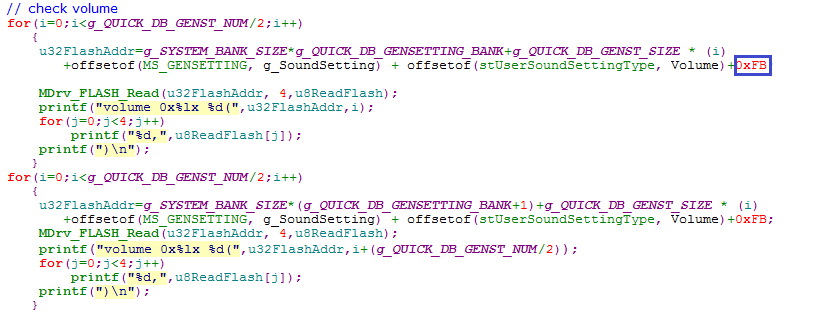
B.RM\_SIZE\_GENSET表示是在DDR里面分配的空间。

C. QUICK\_DB\_GENST\_NUM下面数据为16表示每个BANK能存储8次。

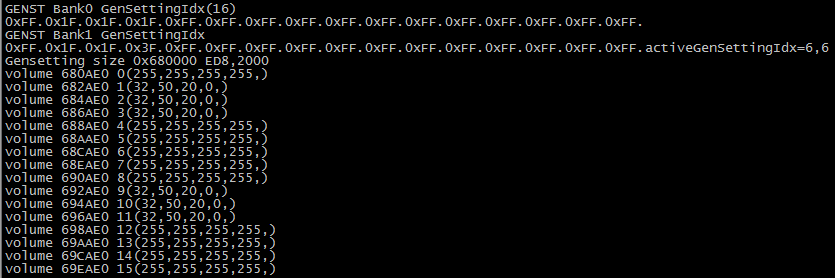




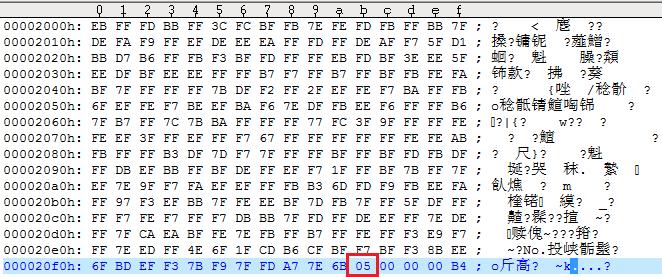
1. **我们先添加函数debugforFlashGenSettingStore（见此文档最后）在MApp\_SaveData.c文件的最下面。**
2. 里面有当前的g\_u16QuickGenSettingIdx号，和FLASH里面有效的g\_u16QuickGenSettingIdx。如果不一样，那说明FLASH一定有存储问题。
3. 还有确认gensetting的分区，看是否有溢出问题。
4. 打印你更改的VOLUME变量在FLASH里面存储的值。
5. 下面确认0XFB是怎么来的？(还可以使用RM\_GENSET\_START\_ADR)



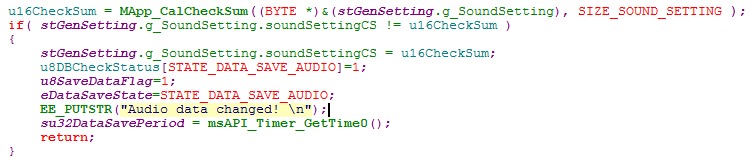
1).把函数添加到软件中，打印如下，我们就知道FLASH中0X680000\0X690000里面就存储了gensetting数据，然后用烧入工具把这段数据读出来。



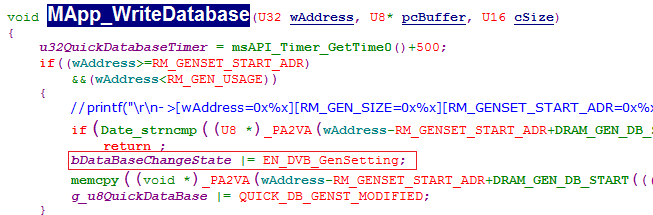
2）因为gensetting里面的第一个变量是u8VersionCheck且变量值为0x05,这样很容易找偏移量是0xFB.



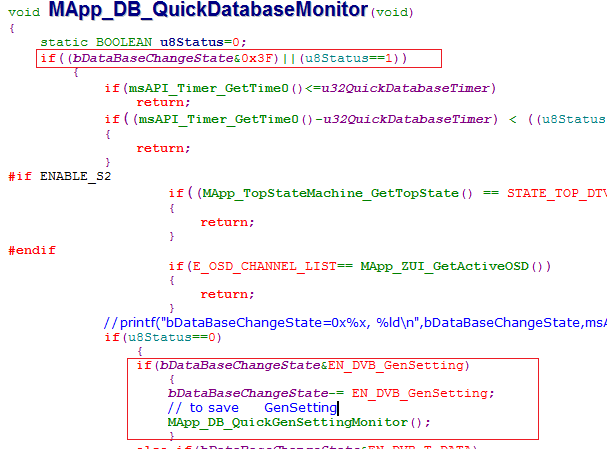
1. **因为上面会把音量值打印出来，每加音量，等几秒打印下，或者用FLASH工具把gensetting数据读出来**。
2. **下面我们一起来看下更改stGenSetting.g\_SoundSetting.Volume后，是怎么保存数据**？
3. 更改数据后，软件会自动去判断soundSettingCS（可以查看MApp\_DataCheckHandler函数）



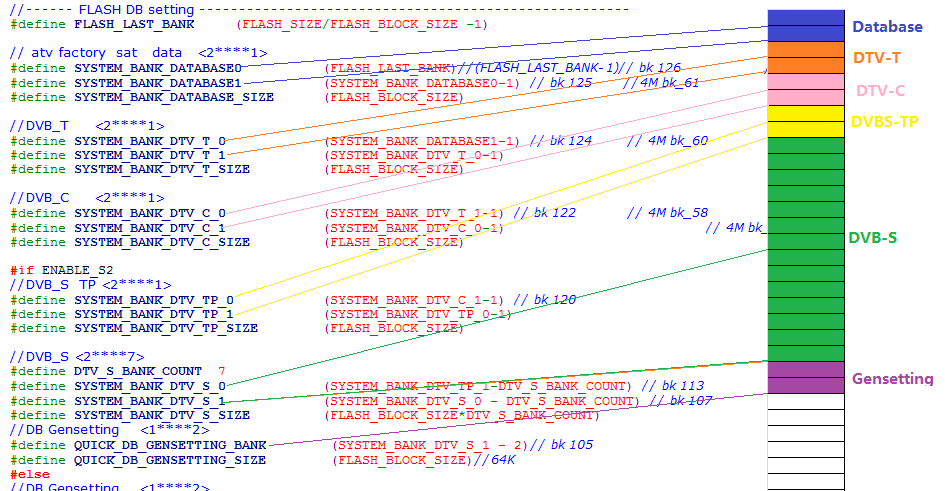
1. 然后u8SaveDataFlag标志成立后，MApp\_DataSaveHandler（）会运行MApp\_SaveSoundSetting();函数。
2. 然后bDataBaseChangeState |= EN\_DVB\_GenSetting;会设置标志。



1. 然后再运行MApp\_DB\_QuickGenSettingMonitor()函数，保存gensetting所有数据。



1. **MApp\_SaveData.h里面，FLASH空间存储分配**



#include <stddef.h>

#include <string.h>

void debugforFlashGenSettingStore(void)

{

U8 i,j;

U8 u8ReadFlash[32];

U32 u32FlashAddr=0;

U8 activeGenSettingIdx=0;

// check gensetting index

if(g\_bGenSettingStoreUseNewMethod)

{

u32FlashAddr=(g\_SYSTEM\_BANK\_SIZE \* g\_QUICK\_DB\_GENSETTING\_BANK);

printf("\nGENST Bank0 GenSettingIdx(%d)\n",g\_QUICK\_DB\_GENST\_NUM);

MDrv\_FLASH\_Read(u32FlashAddr, g\_QUICK\_DB\_GENST\_NUM,u8ReadFlash);

for(i=0;i<g\_QUICK\_DB\_GENST\_NUM;i++)

{

printf("0x%x.",u8ReadFlash[i]);

if(g\_QUICK\_DB\_GENST\_GOOD==u8ReadFlash[i])

{

activeGenSettingIdx = 2\*i - 1;

}

}

u32FlashAddr=(g\_SYSTEM\_BANK\_SIZE \* (g\_QUICK\_DB\_GENSETTING\_BANK+1));

printf("\nGENST Bank1 GenSettingIdx\n");

MDrv\_FLASH\_Read(u32FlashAddr, g\_QUICK\_DB\_GENST\_NUM,u8ReadFlash);

for(i=0;i<g\_QUICK\_DB\_GENST\_NUM;i++)

{

printf("0x%x.",u8ReadFlash[i]);

if(g\_QUICK\_DB\_GENST\_GOOD==u8ReadFlash[i])

{

activeGenSettingIdx = 2\*i;

}

}

}

else

{

u32FlashAddr=(g\_SYSTEM\_BANK\_SIZE \* g\_QUICK\_DB\_GENSETTING\_BANK);

printf("\nGENST Bank0 GenSettingIdx(%d)\n",g\_QUICK\_DB\_GENST\_NUM);

MDrv\_FLASH\_Read(u32FlashAddr, g\_QUICK\_DB\_GENST\_NUM,u8ReadFlash);

for(i=0;i<g\_QUICK\_DB\_GENST\_NUM;i++)

{

printf("0x%x.",u8ReadFlash[i]);

if(g\_QUICK\_DB\_GENST\_GOOD==u8ReadFlash[i])

{

activeGenSettingIdx=i;

}

}

}

printf("activeGenSettingIdx=%d,%d\n",activeGenSettingIdx,g\_u16QuickGenSettingIdx);

// check Gensetting size

if(RM\_SIZE\_GENSET>=((FLASH\_BLOCK\_SIZE\*2)/QUICK\_DB\_GENST\_NUM))

printf("GENSETTING error\n");

printf("Gensetting size 0x%lx %x,%x \n",g\_SYSTEM\_BANK\_SIZE\*g\_QUICK\_DB\_GENSETTING\_BANK,RM\_SIZE\_GENSET,((FLASH\_BLOCK\_SIZE\*2)/QUICK\_DB\_GENST\_NUM));

// check volume

for(i=0;i<g\_QUICK\_DB\_GENST\_NUM/2;i++)

{

u32FlashAddr=g\_SYSTEM\_BANK\_SIZE\*g\_QUICK\_DB\_GENSETTING\_BANK+g\_QUICK\_DB\_GENST\_SIZE \* (i)

+offsetof(MS\_GENSETTING, g\_SoundSetting) + offsetof(stUserSoundSettingType, Volume)+0xFB;

MDrv\_FLASH\_Read(u32FlashAddr, 4,u8ReadFlash);

printf("volume 0x%lx %d(",u32FlashAddr,2\*i - 1);

for(j=0;j<4;j++)

printf("%d,",u8ReadFlash[j]);

printf(")\n");

}

for(i=0;i<g\_QUICK\_DB\_GENST\_NUM/2;i++)

{

u32FlashAddr=g\_SYSTEM\_BANK\_SIZE\*(g\_QUICK\_DB\_GENSETTING\_BANK+1)+g\_QUICK\_DB\_GENST\_SIZE \* (i)

+offsetof(MS\_GENSETTING, g\_SoundSetting) + offsetof(stUserSoundSettingType, Volume)+0xFB;

MDrv\_FLASH\_Read(u32FlashAddr, 4,u8ReadFlash);

printf("volume 0x%lx %d(",u32FlashAddr,2\*i);

for(j=0;j<4;j++)

printf("%d,",u8ReadFlash[j]);

printf(")\n");

}

}