最近有小伙伴问了这样一个问题，说 kea64 的 eeprom 的基地址是 0x1000_0000，但是 FCCOB 的命令，需要的地址是地址的 0-23 位。

Table 18-3. FCCOB – flash and EEPROM command mode typical usage

CCOBIX[2:0]	Byte	FCCOB parameter fields in flash and EEPROM command mode
000	HI	FCMD[7:0] defining flash command
	LO	Global address [23:16]
001	HI	Global address [15:8]
	LO	Global address [7:0]
010	HI	Data 0 [15:8]
	LO	Data 0 [7:0]
011	HI	Data 1 [15:8]
	LO	Data 1 [7:0]
100	HI	Data 2 [15:8]
	LO	Data 2 [7:0]
101	HI	Data 3 [15:8]
	LO	Data 3 [7:0]

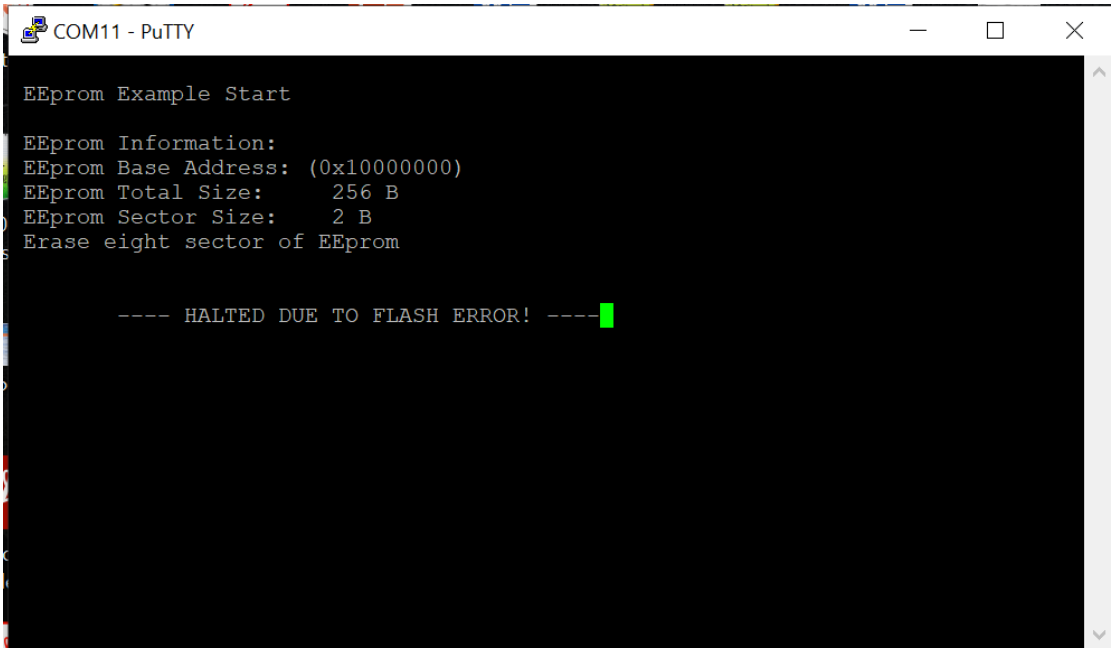
那我们基地址的 1，不就不会被放入 FCCOB 里吗？因为 1 是在第 28 位啊。
既然地址都没完整放进去，那怎么擦除 eeprom 指定地址呢？
事实上，并不需要完整的地址，mcu 可以根据 FCCOB 的命令域是 flash 命令还是 eeprom 命令来自动明白从哪个基地址开始操作。也就是说，当命令域是 eeprom 的擦写等操作时候，mcu 就自动明白会从 0x1000_0000 开始进行操作。为了验证这一点，我们来做一个实验。
我使用了 FRDM-KE02 的 eeprom 例程来实验，它的基地址也是 0x1000_0000。
源代码中要擦写 eeprom 的目标地址是

```
destAdrss = EEpromBlockBase +
                (EEpromTotalSize - (SECTOR_INDEX_FROM_END * EEpromSectorSize))
```

我们现在将 eeprom 的基地址 EEpromBlockBase 设置为 0。如果没有基地址，还能操作 eeprom，那就说明我们是对的。destAdrss 现在值为 0xf0

```
*/
#ifndef SECTOR_INDEX_FROM_END
#define SECTOR_INDEX_FROM_END 8U
#endif

  /* Erase a sector from destAdrss. */
//  destAdrss = EEpromBlockBase +
//              (EEpromTotalSize - (SECTOR_INDEX_FROM_END * EEpromSectorSize));
   destAdrss = 0 +
               (EEpromTotalSize - (SECTOR_INDEX_FROM_END * EEpromSectorSize));

  /*! Eeprom can be written directly by running the FLASH_EepromWrite function,
  Execute this erase function ensures that the eeprom is in the erased state*/
  result = FLASH_EraseEEprom(&s_flashDriver, destAdrss, sizeof(s_buffer),
                             kFLASH_ApiEraseKey);
  if (kStatus_FLASH_Success != result) {
    error_trap();
  }
  /* Prepare user buffer. */
```

然后运行一下



程序挂了，难道思路不对吗?

仔细debug程序,会发现,我们需要注释这段代码,它做了地址范围检查,检测在不在eeprom地址范围里，我们现在测试的地址当然不在，所以我们暂时注释掉。

```
#if FLASH_SSD_IS_EEPROM_ENABLED
status_t FLASH_EraseEEprom(flash_config_t *config, uint32_t start,
                           uint32_t lengthInBytes, uint32_t key) {
  uint32_t sectorSize;
  flash_operation_config_t flashOperationInfo;
  uint32_t endAddress;      /* storing end address */
  uint32_t numberOfSectors; /* number of sectors calculated by endAddress */
  status_t returnCode;

  flash_get_matched_operation_info(config, start, &flashOperationInfo);

  /* Check the supplied address range. */
//  returnCode = EEPROM_check_range(config, start, lengthInBytes,
//                                  config->EEpromSectorSize);
//  if (returnCode) {
//    return returnCode;
//  }
  /* Validate the user key */
  returnCode = flash_check_user_key(key);
  if (returnCode) {
    return returnCode;
  }
```

这个擦除 eeprom 需要暂时注释，同样的写入 eeprom 的函数里有相同的关于地址范围检测的代码，我们暂时也注释掉。

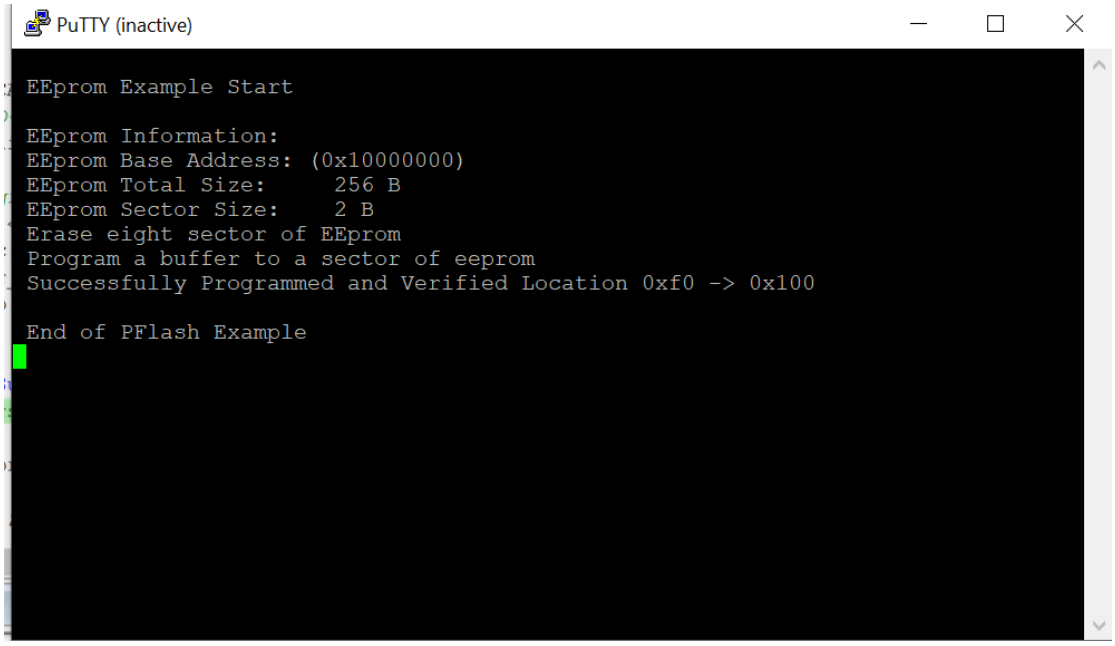
```
#if FLASH_SSD_IS_EEPROM_ENABLED
status_t FLASH_EepromWrite(flash_config_t *config, uint32_t start, uint8_t *src,
                           uint32_t lengthInBytes) {
  status_t returnCode;
  uint32_t i;
  if ((config == NULL) || (src == NULL)) {
    return kStatus_FLASH_InvalidArgument;
  }
  if ((lengthInBytes > 4) || (0 == lengthInBytes)) {
    return kStatus_FLASH_InvalidArgument;
  }

  /* Check the supplied address range. */
  /* Validates the range of the given address */
//  if ((start < config->EEpromBlockBase) ||
//      ((start + lengthInBytes) >
//       (config->EEpromBlockBase + config->EEpromTotalSize))) {
//    return kStatus_FLASH_AddressError;
//  }

  returnCode = kStatus_FLASH_Success;

  flash_cache_clear_process(config, kFLASH_CacheClearProcessPre);
```

同时我们还要修改读取地址内容的代码，虽然我们是将 destAdrss 的基地址设置为 0 了，但是读取时候还是要从 eeprom 位置开始读。

```
  }
  /* Verify erase by reading back from eeprom directly*/
  for (i = 0; i < BUFFER_LEN; i++) {
    s_buffer_rbc[i] = *(volatile uint8_t *)(EEpromBlockBase + destAdrss + i);
    if (s_buffer_rbc[i] != s_buffer[i]) {
      error_trap();
    }
  }
  /* Print message for user. */
```

```
#endif
  /* Verify programming by reading back from eeprom directly*/
  for (i = 0; i < BUFFER_LEN; i++) {
    s_buffer_rbc[i] = *(volatile uint8_t *)(EEpromBlockBase + destAdrss + i);
    if (s_buffer_rbc[i] != s_buffer[i]) {
      error_trap();
    }
  }
```

改完以后运行一下，发现通过了



我们提供的地址里去除了 eeprom 的基地址是 0xf0，但是它仍然能修改和读取 eeprom 的数

据，所以其实基地址并不需要去指定，mcu 自己明白 eeprom 基地址在哪里