Technical Note

Enabling SD/uSD Card Lock/Unlock Feature in Linux®

Introduction

The lock/unlock feature of SD/uSD cards enables the host system, using a security password, to prevent illegal or unauthorized access to user data. This technical note systematically describes the lock/unlock operating sequence that conforms to the SD Association Memory Card Specification version 2.00, which made the various password lock/unlock commands mandatory when they were optional in previous versions of the specification. The following sections explain how to enable the lock/unlock feature using the Linux® mmc-utils with sample source code as reference.

Lock/Unlock Feature

SD/uSD card lock/unlock operations enable the host to lock or unlock a card with a 16-byte maximum length password. CMD42 (LOCK_UNLOCK) is used lock/unlock and has the same structure and bus transaction type as regular single-block WRITE command. If the card is locked after it has been selected using CMD7, the host unlocks it with the correct password before sending other unsupported commands.

The following table describes the structure of the command data block, which includes all of the required information (operation mode, password length and password data itself).

Table 1: Lock/Unlock Command Data Structure

Command	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved (shall be set to 0)				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN (password length)							
2	Password data							
...								
PWDS_LEN + 1								

The password and its size are kept in two nonvolatile registers, the 128-bit password data register (PWD) and the 8-bit password length register (PWDS_LEN), respectively. As a result, the lock/unlock state is preserved and cannot be erased by power cycling. PWDS_LEN indicates whether a password is currently set. When it equals 0, a password has not been set. If the password was previously set (the value of PWDS_LEN is not 0), the card will be locked automatically after power-on.

The data block size is defined by the host before it sends the card lock/unlock command and will be set to greater than or equal to the required data structure of the LOCK_UNLOCK command. The data block size is specified by CMD16, and the block length can be up to 512 bytes for the card lock/unlock command.

Operation mode will be selected every time the lock/unlock command is sent. The mode options are stored in bits 0 to 3 of byte 0 in the data block (bits 4 to 7 will be set to 0). The detail command sequence of each mode are discussed in the following sections.

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.
Products and specifications discussed herein are subject to change by Micron without notice.

Password Management

Set Password

If the password has not been set previously or was cleared by CLR_PWD, a password can be set. To set the password, the host issues CMD42 with a predefined data block size on the data line that includes the 16-bit CRC. The mode (SET_PWD) will be indicated with the password length (PWDS_LEN) and the password itself in the data block. The password may be up to 16 bytes in length.

When a password is replaced, the length value (PWDS_LEN) will include both new and old (currently used) passwords with a maximum length of 32 bytes. Likewise, the password data field will include the old password followed by the new password. The card is able to identify the new password field range internally through calculation of the new password length by subtracting the old password length from the PWDS_LEN field. The new password is then given and its size saved in the PWD and PWDS_LEN registers, respectively, after the successful execution of CMD42.

If the old password given is incorrect (not equal in length and content), the LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password will not change. If the password length is 0 or greater than 128 bits, the card will indicate an error with LOCK_UNLOCK_FAILED in the status register.

Clear Password

The procedure for clearing a password is much the same as the one used to set a password: the host issues CMD42 with a predefined data block size on the data line that includes the 16-bit CRC. The mode (CLR_PWD) will be indicated with the password length (PWDS_LEN) and the password itself in the data block. If PWD and PWDS_LEN match the sent password and its length, the content of the PWD register is cleared and PWDS_LEN is set to 0. If the password is not correct, the LOCK_UNLOCK_FAILED error bit will be set in the status register.

LOCK/UNLOCK Operation

Lock the Card

To lock the card, the host issues CMD42 with a predefined data block size on the data line that includes the 16-bit CRC. The mode (LOCK) is indicated with the password length (PWDS_LEN) and the password itself in the data block. If the PWD content is equal to the sent password, the card will be locked immediately in the current power session and the CARD_IS_LOCKED bit in the status register will be set to indicate the card has locked successfully.

If the sent password is not equal to the sent password, the LOCK_UNLOCK_FAILED error bit will be set in the status register. An attempt to lock a locked card or to lock a card that does not have a password will lead to the LOCK_UNLOCK_FAILED error bit set in the status register.

Operations to set the password and lock the card can be combined in the same process. In this case, the host performs all of the required steps for setting the password (as described in Set Password (page 2)) including the bit LOCK set while CMD42 is being sent.

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

The locked card accepts the following commands and returns responses by setting CARD_IS_LOCKED. All other commands are treated as illegal commands.

- Basic class (0)
- Lock card class (7)
- CMD16
- ACMD41
- ACMD42

Unlock the Card

To unlock the card, the host issues CMD42 with a predefined data block size on the data line that includes the 16-bit CRC. The mode (UNLOCK) is indicated with the password length (PWDS_LEN) and the password itself in the data block.

If the PWD content is equal to the sent password, the card will be unlocked and the CARD_IS_LOCKED bit in the status register will be cleared. Note that unlocking is done only for the current power session and the card will go back to a lock state automatically on the next power-up as long as the password is not cleared in prior power cycle.

The LOCK_UNLOCK_FAILED error bit in the status register will be set if the password is not matched. An attempt to unlock an unlocked card will fail and the LOCK_UNLOCK_FAILED error bit in the status register will be set.

Force Erase

When a user forgets the password (PWD content), it is possible to erase PWD content and get back to an unlocked state using ERASE mode in CMD42. However, **all of the card content** including the PWD and PWDS_LEN register content will be completely erased for security reasons. This operation is called FORCE ERASE.

To implement A FORCE ERASE operation, the first byte of the CMD42 data block is enough for mode selection. Therefore, the block length can be defined to one byte by CMD16 in SDR mode. For DDR50 mode, because the block length will always be even, the block length for the FORCE ERASE command will be set to an even number (at least 2) as well. After the block length is set, CMD42 is issued with a predefined data block size on the data line that includes the 16-bit CRC; the mode ERASE will be the only bit set in the data block.

If the ERASE bit is not the only bit set in the data field, the LOCK_UNLOCK_FAILED error bit in the status register will be set and the erase request is rejected. An attempt to force erase on an unlocked card will fail and the LOCK_UNLOCK_FAILED error bit will be set in the status register.

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

CMD42 Parameter and Result

To use CMD42 correctly, the host must guarantee all operations conform to the settings and sequences detailed in previous sections. For a comprehensive working scenario and result output, refer to the following table for a detail check.

Table 2: Lock/Unlock Command Truth Table

Lock/Unlock Mode				Current State	PWDS_LEN and PWD	Operation Result	Card Status	
Bit3	Bit2	Bit1	Bit0				Bit25[1]	Bit24[2]
1	0	0	0	Locked	Exist	Force Erase	1 to 0	0
1	0	0	0	Unlocked	Exist	Error	0	1
1	0	0	0	Unlocked	Cleared	Error	0	1
0	1	0	0	Locked	Exist	Error	1	1
0	1	0	0	Unlocked	Exist	Lock the card	0 to 1	0
0	1	0	0	Unlocked	Cleared	Error	0	1
0	1	0	1	Locked	Exist	Replace password and the card is still locked	1	0
0	1	0	1	Unlocked	Exist	Replace password and the card is locked	0 to 1	0
0	1	0	1	Unlocked	Cleared	Set password and lock the card	0 to 1	0
0	0	1	0	Locked	Exist	Clear PWDS_LEN and PWD, the card is unlocked	1 to 0	0
0	0	1	0	Unlocked	Exist	Clear PWDS_LEN and PWD	0	0
0	0	1	0	Unlocked	Cleared	Error	0	1
0	0	0	1	Locked	Exist	Replace password and the card is unlocked	1 to 0	0
0	0	0	1	Unlocked	Exist	Replace password and the card is unlocked	0	0
0	0	0	1	Unlocked	Cleared	Set password and the card is still unlocked	0	0
0	0	0	0	Locked	Exist	Unlock the card	1 to 0	0
0	0	0	0	Unlocked	Exist	Error	0	1
0	0	0	0	Unlocked	Cleared	Error	0	1
After power-on					Exist	The card is locked	1	0
					Cleared	The card is unlocked	0	0
Other combinations				Don't Care	Don't Care	Error	0 or 1	1

Notes:
1. CARD_IS_LOCKED (bit 25 of the card status register) can be seen in the response of CMD13 after CMD42.
2. LOCK_UNLOCK_FAILED (bit 24 of the card status register) as the result of CMD42 can be seen in the response of either CMD42 or the following CMD13.

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

Linux Environment Requirements and Setup

Micron provides demo source code leveraging MMC/SD ioctl device interface (mmc-utils) to implement all lock/unlock related functions. Linux mmc-utils is an open source user-space test tool for MMC/SD devices. The installation and compilation steps are as follows.

1. Download the latest mmc_utils distribution from,

```
  $> sudo git clone https://kernel.googlesource.com/pub/scm/
linux/kernel/git/cjb/
```

2. Build the sources using the make build utility with appropriate cross-compiler.

3. Check the availability of mmc utilities tool by,

```
  $> ./mmc -h
```

Lock/Unlock Functions Linux Porting

Four source files must be modified to add CMD42 security feature support in mmc-utils standard distribution.

Change in mmc-utils/mmc.h

mmc-utils/mmc.h includes a list of all global variables used in mmc-utils. The following definitions must be created for lock/unlock feature support.

```
#define MMC_SET_BLOCKLEN        16  /* ac [31:0] block len R1 */
#define MMC_LOCK_UNLOCK         42  /* adtc R1b */
#define MMC_CMD42_UNLOCK        0x0 /* UNLOCK */
#define MMC_CMD42_SET_PWD       0x1 /* SET_PWD */
#define MMC_CMD42_CLR_PWD       0x2 /* CLR_PWD */
#define MMC_CMD42_LOCK          0x4 /* LOCK */
#define MMC_CMD42_SET_LOCK      0x5 /* SET_PWD & LOCK */
#define MMC_CMD42_ERASE         0x8 /* ERASE */
#define MAX_PWD_LENGTH          32  /* max PWDS_LEN: old+new */
#define MMC_BLOCK_SIZE          512 /* data blk size for cmd42 */
#define MMC_R1_ERROR            (1 << 19) /* R1 bit19 */
#define MMC_R1_LOCK_ULOCK_FAIL  (1 << 24) /* R1 bit24 */
```

Change in mmc-utils/mmc.c

The command-line user interface (CLI) details are maintained in mmc-utils/mmc.c. CMD42 is defined in the commands structure.

```
{ do_lock_unlock, -3,
    "cmd42", "<password> " "<parameter> " "<device>\n"
    "Usage: mmc cmd42 <password> <s|c|l|u|e> <device>\n"
    "s\tset password\n" "c\tclear password\n" "l\tlock\n" "sl
\tset password and lock\n" "u\tunlock\n" "e\tforce erase\n",
    NULL
},
```

Change in mmc-utils/mmc_cmds.h

mmc-utils/mmc_cmds.h includes a set of CLI command declarations. These CLI command wrappers are used to encapsulate specific command behavior from the lower level mmc ioctl system calls. do_lock_unlock(int nargs, char **argv) and set_block_len(int

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

fd, int blk_len) in mmc_cmds.c require corresponding matching function declarations in mmc_cmds.h.

```
int do_lock_unlock(int nargs, char **argv);
int set_block_len(int fd, int blk_len);
```

Change in mmc-utils/mmc_cmds.c

mmc-utils/mmc_cmds.c has CLI command wrapper for each function declared in mmc-utils/mmc_cmds.h. These functions will send custom commands to the card by using ioctl(fd, MMC_IOC_CMD, (struct mmc_ioc_cmd*) &ioctl_data) with fd pointing to correct mmcblkn device. do_lock_unlock(int nargs, char **argv) and set_block_len(int fd, int blk_len) are added in mmc_cmds.c for lock/unlock feature enablement.

```
//lock/unlock feature implementation
int do_lock_unlock(int nargs, char **argv)
{
    int fd, ret = 0;
    char *device;
    __u8 data_block[MMC_BLOCK_SIZE]={0};
    __u8 data_block_onebyte[1]={0};
    int block_size = 0;
    struct mmc_ioc_cmd idata;
    int cmd42_para;             //parameter of cmd42
    char pwd[MAX_PWD_LENGTH+1];      //password
    int pwd_len;                     //password length
    __u32 r1_response;               //R1 response token

    CHECK(nargs != 4, "Usage: mmc cmd42 <password> <s|c|l|u|e>
<device>\n",
              exit(1));

    strcpy(pwd, argv[1]);
    pwd_len = strlen(pwd);

    if (!strcmp("s", argv[2])) {
        cmd42_para = MMC_CMD42_SET_PWD;
        printf("Set password: password=%s ...\n", pwd);
    }
    else if (!strcmp("c", argv[2])) {
        cmd42_para = MMC_CMD42_CLR_PWD;
        printf("Clear password: password=%s ...\n", pwd);
    }
    else if (!strcmp("l", argv[2])) {
        cmd42_para = MMC_CMD42_LOCK;
        printf("Lock the card: password=%s ...\n", pwd);
    }
    else if (!strcmp("sl", argv[2])) {
        cmd42_para = MMC_CMD42_SET_LOCK;
        printf("Set password and lock the card: password - %s ...
\n", pwd);
    }
    else if (!strcmp("u", argv[2])) {
        cmd42_para = MMC_CMD42_UNLOCK;
        printf("Unlock the card: password=%s ...\n", pwd);
    }
    else if (!strcmp("e", argv[2])) {
        cmd42_para = MMC_CMD42_ERASE;
```


Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

```
        printf("Force erase ... (Warning: all card data will be
erased together with PWD!)\n");
    }
    else {
        printf("Invalid parameter:\n" "s\tset password\n" "c
\tclear password\n" "l\tlock\n"
               "sl\tset password and lock\n" "u\tunlock\n" "e
\tforce erase\n");
        exit(1);
    }

    device = argv[nargs-1];

    fd = open(device, O_RDWR);
    if (fd < 0) {
        perror("open");
        exit(1);
    }

    if (cmd42_para==MMC_CMD42_ERASE)
        block_size = 2; //set  blk size to 2-byte for Force
Erase @DDR50 compability
    else
        block_size = MMC_BLOCK_SIZE;

    ret = set_block_len(fd, block_size);  //set data block size
prior to cmd42
        printf("Set to data block length = %d byte(s).\n",
block_size);

    if (cmd42_para==MMC_CMD42_ERASE) {
        data_block_onebyte[0] = cmd42_para;
    } else {
        data_block[0] = cmd42_para;
        data_block[1] = pwd_len;
        memcpy((char *)(data_block+2), pwd, pwd_len);
    }

    memset(&idata, 0, sizeof(idata));
    idata.write_flag = 1;
    idata.opcode = MMC_LOCK_UNLOCK;
    idata.arg = 0;        //set all 0 for cmd42 arg
    idata.flags = MMC_RSP_R1 | MMC_CMD_AC | MMC_CMD_ADTC;
    idata.blksz = block_size;
    idata.blocks = 1;

    if (cmd42_para==MMC_CMD42_ERASE)
        mmc_ioc_cmd_set_data(idata, data_block_onebyte);
    else
        mmc_ioc_cmd_set_data(idata, data_block);

    ret = ioctl(fd, MMC_IOC_CMD, &idata);   //Issue CMD42

    r1_response = idata.response[0];
    printf("cmd42 response: 0x%08x\n", r1_response);
    if (r1_response & MMC_R1_ERROR) {     //check CMD42 error
        printf("cmd42 error! Error code: 0x%08x\n", r1_response
& MMC_R1_ERROR);
```


Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

```
        ret=-1;
    }
    if (r1_response & MMC_R1_LOCK_ULOCK_FAIL) {     //check lock/
unlock error
        printf("Card lock/unlock fail! Error code: 0x%08x\n",
r1_response & MMC_R1_LOCK_ULOCK_FAIL);
        ret=-1;
    }

    close(fd);
    return ret;
}

//change data block length
int set_block_len(int fd, int blk_len)
{
    int ret = 0;
    struct mmc_ioc_cmd idata;

    memset(&idata, 0, sizeof(idata));
    idata.opcode = MMC_SET_BLOCKLEN;
    idata.arg = blk_len;
    idata.flags = MMC_RSP_R1 | MMC_CMD_AC;

    ret = ioctl(fd, MMC_IOC_CMD, &idata);

    return ret;
}
```

Demo Test Example

To check the arguments of CMD42 in mmc-utils, use '--help' option under CLI terminal.

```
$> ./mmc cmd42 --help
   Usage:
        mmc cmd42 <password> <parameter> <device>
                Usage: mmc cmd42 <password> <s|c|l|u|e> <device>
                s       set password
                c       clear password
                l       lock
                sl      set password and lock
                u       unlock
                e       force erase
```

A series of interactive operations in the testing sequence below demonstrates the use of CMD42 with the combination of all supported parameter mode settings:

- Set password to 'old_pwd'
- Replace password 'old_pwd' with 'new_pwd'
- Lock the card, check status
- Unlock the card, check status
- Clear password
- Set password to 'pwd' and lock the card simultaneously, check status

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

- Force Erase the card, check status

```
$> ./mmc cmd42 old_pwd s /dev/mmcblk1
   Set password: password=old_pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x00000900
$> ./mmc cmd42 old_pwdnew_pwd s /dev/mmcblk1
   Set password: password=old_pwdnew_pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x00000900
$> ./mmc cmd42 new_pwd l /dev/mmcblk1
   Lock the card: password=new_pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x00000900
$> ./mmc status get /dev/mmcblk1
   SEND_STATUS response: 0x02000900
$> ./mmc cmd42 new_pwd u /dev/mmcblk1
   Unlock the card: password=new_pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x02000900
$> ./mmc status get /dev/mmcblk1
   SEND_STATUS response: 0x00000900
$> ./mmc cmd42 new_pwd c /dev/mmcblk1
   Clear password: password=new_pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x00000900
$> ./mmc cmd42 pwd sl /dev/mmcblk1
   Set password and lock the card: password - pwd ...
   Set to data block length = 512 byte(s)
   cmd42 response: 0x00000900
$> ./mmc status get /dev/mmcblk1
   SEND_STATUS response: 0x02000900
$> ./mmc cmd42 forget e /dev/mmcblk1
   Force erase ... (Warning: all card data will be erased togeth-
er with PWD!)
   Set to data block length = 1 byte(s)
   cmd42 response: 0x02000900
$> ./mmc status get /dev/mmcblk1
   SEND_STATUS response: 0x00000900
```


Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.

Revision History

Rev. A – 8/17

- Initial release

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-4000
www.micron.com/products/support Sales inquiries: 800-932-4992
Micron and the Micron logo are trademarks of Micron Technology, Inc.
All other trademarks are the property of their respective owners.
This data sheet contains minimum and maximum limits specified over the power supply and temperature range set forth herein.
Although considered final, these specifications are subject to change, as further product development and data characterization sometimes occur.

Micron Technology, Inc. reserves the right to change products or specifications without notice.
© 2017 Micron Technology, Inc. All rights reserved.