Technical Note

Enabling Micron Memory Card Health Monitor System

Introduction

MicroSD and SD card product lifespan depends primarily on the reliability of the product's NAND technology. Host systems are recommended to periodically monitor the NAND health status to ensure the device is not approaching end of life. Continuous NAND memory card use in application systems has an endurance limit after which data integrity can be compromised. The HEALTH STATUS command enables a host to monitor the total NAND write percentage to determine whether the card is approaching prescribed limits.

This technical note describes the HEALTH STATUS command functionality tailored for Micron® industrial memory card products. Included is sample code providing a step by step guide for users to enable the HEALTH STATUS command in Linux® and non-Linux application systems.

NOTE: This document is in reference to Micron® i200, i300, and i400 family industrial microSD and SD card products.

The HEALTH STATUS command in a host system enables users to monitor media usage throughout a memory card's remaining cycle time and take corrective action before an endurance limit. This is especially helpful to applications where heavy usage stresses NAND cell endurance.

Micron recognizes the value of open source compatibility and the importance of supporting the open source community. The sample source code in this document can be used as a reference for non-Linux operating system software development. For additional support to enable the HEALTH STATUS command for Micron high endurance microSD and SD cards, contact your Micron representative.

Important Notes and Warnings

Micron Technology, Inc. ("Micron") reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions. This document supersedes and replaces all information supplied prior to the publication hereof. You may not rely on any information set forth in this document if you obtain the product described herein from any unauthorized distributor or other source not authorized by Micron.

Automotive Applications. Products are not designed or intended for use in automotive applications unless specifically designated by Micron as automotive-grade by their respective data sheets. Distributor and customer/distributor shall assume the sole risk and liability for and shall indemnify and hold Micron harmless against all claims, costs, damages, and expenses and reasonable attorneys' fees arising out of, directly or indirectly, any claim of product liability, personal injury, death, or property damage resulting directly or indirectly from any use of non-automotive-grade products in automotive applications. Customer/distributor shall ensure that the terms and conditions of sale between customer/distributor and any customer of distributor/customer (1) state that Micron products are not designed or intended for use in automotive applications unless specifically designated by Micron as automotive-grade by their respective data sheets and (2) require such customer of distributor/customer to indemnify and hold Micron harmless against all claims, costs, damages, and expenses and reasonable attorneys' fees arising out of, directly or indirectly, any claim of product liability, personal injury, death, or property damage resulting from any use of non-automotive-grade products in automotive applications.

Critical Applications. Products are not authorized for use in applications in which failure of the Micron component could result, directly or indirectly in death, personal injury, or severe property or environmental damage ("Critical Applications"). Customer must protect against death, personal injury, and severe property and environmental damage by incorporating safety design measures into customer's applications to ensure that failure of the Micron component will not result in such harms. Should customer or distributor purchase, use, or sell any Micron component for any critical application, customer and distributor shall indemnify and hold harmless Micron and its subsidiaries, subcontractors, and affiliates and the directors, officers, and employees of each against all claims, costs, damages, and expenses and reasonable attorneys' fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such critical application, whether or not Micron or its subsidiaries, subcontractors, or affiliates were negligent in the design, manufacture, or warning of the Micron product.

Customer Responsibility. Customers are responsible for the design, manufacture, and operation of their systems, applications, and products using Micron products. ALL SEMICONDUCTOR PRODUCTS HAVE INHERENT FAILURE RATES AND LIMITED USEFUL LIVES. IT IS THE CUSTOMER'S SOLE RESPONSIBILITY TO DETERMINE WHETHER THE MICRON PRODUCT IS SUITABLE AND FIT FOR THE CUSTOMER'S SYSTEM, APPLICATION, OR PRODUCT. Customers must ensure that adequate design, manufacturing, and operating safeguards are included in customer's applications and products to eliminate the risk that personal injury, death, or severe property or environmental damages will result from failure of any semiconductor component.

Limited Warranty. In no event shall Micron be liable for any indirect, incidental, punitive, special or consequential damages (including without limitation lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort, warranty, breach of contract or other legal theory, unless explicitly stated in a written agreement executed by Micron's duly authorized representative.

Command Function and Format

The Micron device HEALTH STATUS protocol, with its specific command argument and response format shown here, extends the SD card generic command and enables host access to device health status. The host can issue the HEALTH STATUS command (CMD56) multiple times during the same power-on cycle.

Table 1: Health Status Command

Command[1]	Input		Output[3]		Description
	Argument[2]	Value	Argument	Value	
CMD56	Bit[31:0] :	11h 00h 05h FBh	Offset[0:3] :	Fixed field header, expected values: 4Dh, 45h, 42h, 55h	First four output bytes must match expected values: only then is data block valid, enabling users to extract health information in subsequent data block output fields.
			Offset[7] :	1% (predefined)	Defines resolution of TLC/QLC and SLC P/E utilization in HEALTH STATUS command.
			Offset[8] :	Hex value of percentage usage (See table below)	TLC/QLC percentage utilizations acquired from data block offset[8]. User data is stored in TLC/QLC NAND area.
			Offset[9] :		SLC percentage utilizations acquired from data block offset[9]. FW system blocks and some internal cache buffers are deployed to SLC configured range.
			Others	—	—

Notes: 1. CMD56 (GEN_CMD) with Micron specific details: command argument, command in-bytes data block structure. After the host receives a valid R1 response from the SD card, it then gets the health information report from the data block returned by CMD56. After completing the transfer, the SD card returns to the Transfer State.
2. Bit[0] in the argument is Read or Write mode selection bit. A value of '1' is required to indicate the HEALTH STATUS command is working under Read mode.
3. Block length is fixed to 512 bytes for SDHC and SDXC.

Table 2: Hex Value of Percentage Usage from Offset[8] and Offset[9]

Hex Value	01h	02h	03h	...	0Ah	...	32h	...	5Ah	...	63h	64h
% Used	1%	2%	3%	...	10%	...	50%	...	90%	...	99%	100%

Table 3: HEALTH STATUS Information Output Data Example from Used Card

Offset	Output Data (512 Bytes)	
000	4D 45 42 55 FF FF FF 01	15 02 FF FF FF FF FF FF
016	FF FF FF FF FF FF FF FF	FF FF FF FF FF FF FF FF
032	FF FF FF FF FF FF FF FF	FF FF FF FF FF FF FF FF
...	...	...
480	FF FF FF FF FF FF FF FF	FF FF FF FF FF FF FF FF
496	FF FF FF FF FF FF FF FF	FF FF FF FF FF FF FF FF

The following applies for this example:

- Offset[0:3] = 4Dh 45h 42h 55h: the field header.
- Offset[7] = 01h: percentage step is 1%.
- Offset[8] = 15h: TLC/QLC utilization is 21%.
- Offset[9] = 02h: SLC utilization is 2%.
- Other bits are stuffed with FFh.

Linux Environment Setup

To help users implement the HEALTH STATUS command, Micron provides sample source code leveraging the MMC/SD ioctl device interface (mmc-utils). Linux mmc-utils is an open source user-space test tool for MMC/SD devices.

Before implementing the Linux porting instructions in this document, users must perform the following installation and compilation steps to set up a working environment under Linux.

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
```

Health Status Command Linux Porting

The four source files below must be modified to add the HEALTH STATUS command support in mmc-utils standard distribution.

Table 4: Changes in mmc-utils Files

File	Includes	Modification
mmc-utils/ mmc.h	List of all global variables used in mmc-utils.	Create definitions for the HEALTH STATUS command support in the mmc.h file.
mmc-utils/ mmc.c	Command-line user interface (CLI) details.	Create the command structure for the HEALTH STATUS command in the mmc.c file.
mmc-utils/ mmc_cmds.h	Set of CLI command declarations: These CLI command wrappers are used to encapsulate specific command behavior from the lower level mmc ioctl system calls. These functions send custom commands to the card by using ioctl(fd, MMC_IOC_CMD, (struct mmc_ioc_cmd*) &ioctl_data) with fd pointing to the correct mmcblkn device.	From mmc_cmds.c, create the following corresponding matching function declarations in the mmc_cmds.h file:
		Main function(to implement the HEALTH STATUS command)**:** int do_PPEU(int nargs, char **argv);
		Sub-function: int CMD56_data_in(int fd, int cmd56_arg, char *lba_block_data);
		Sub-function: void dump_data_block(char *lba_block_data);
mmc-utils/ mmc_cmds.c	CLI command wrappers for each function declared in mmc-utils/mmc_cmds.h.	In the mmc_cmds.c file, create CLI command wrappers for each function declared in mmc-utils/mmc_cmds.h file as listed above.

Note: 1. Applies to entire table: See also code block tables that follow.

Table 5: mmc.h File

```
#define SD_GEN_CMD 56/* adtc, R1 */
#define SD_BLOCK_SIZE 512   /* data block size for CMD56 */
#define MMC_RSP_R2    (MMC_RSP_PRESENT|MMC_RSP_136|MMC_RSP_CRC)
#define MMC_CMD_BCR   (3 << 5)
```

Table 6: mmc.c File

```
{ do_PPEU, -1,
    "ppeu read", "<dump ctrl> <device>\n"
    "Usage: mmc ppeu read <-d> <device>\n"
    "-d\tdump data block",
    NULL
},
```

Table 7: mmc_cmds.h File

```
int do_PPEU(int nargs, char **argv);/* Main function
int CMD56_data_in(int fd, int cmd56_arg, char *lba_block_data);/* Sub-function
void dump_data_block(char *lba_block_data);/* Sub-function
```

Table 8: mmc_cmds.c File—do_PPEU Main Function

```
//PPEU - Percentage of P/E cycles Used
int do_PPEU(int nargs, char **argv)
{
int cmd56_arg = 0x110005FB;
char data_in[SD_BLOCK_SIZE];
int fd, ret;
char *device;

CHECK(!((nargs == 2)||(nargs == 3)),
"Usage: mmc ppeu read <-d> <device>\n", exit(1));
device = argv[nargs-1];
fd = open(device, O_RDWR);
if (fd < 0) {
perror("open");
exit(1);
}

//execute CMD56 and get one 512-byte data block
ret = CMD56_data_in(fd, cmd56_arg, data_in);

if (ret) {
fprintf(stderr, "CMD56 CALL FAILED, %s\n", device);
exit(1);
}

if (!strcmp("-d", argv[1]))
    dump_data_block(data_in); //data block dumping

/* write data to stdout in CSV format */
printf("Fixed Filed Header: %02Xh %02Xh %02Xh %02Xh\n",
data_in[0], data_in[1], data_in[2], data_in[3]);
printf("Percentage Step Size: %d\n", data_in[7]);
printf("TLC/QLC Percentage Utilization: %d%%\n", data_in[8]);
printf("SLC Percentage Utilization: %d%%\n", data_in[9]);

return ret;
}
```

Table 9: mmc_cmds.c File—CMD56_data_in Sub-Function

```
//CMD56 implementation
int CMD56_data_in(int fd, int cmd56_arg, char *lba_block_data)
{
int ret = 0;
struct mmc_ioc_cmd idata;

memset(&idata, 0, sizeof(idata));
memset(lba_block_data, 0, sizeof(__u8) * 512);
idata.write_flag = 0;
idata.opcode = SD_GEN_CMD;
idata.arg = cmd56_arg;
idata.flags = MMC_RSP_SPI_R1 | MMC_RSP_R1 | MMC_CMD_ADTC;
idata.blksz = SD_BLOCK_SIZE;
idata.blocks = 1;
mmc_ioc_cmd_set_data(idata, lba_block_data);

ret = ioctl(fd, MMC_IOC_CMD, &idata);
if (ret)
perror("ioctl");

return ret;
}
```

Table 10: mmc_cmds.c File—dump_data_block Sub-Function

```
void dump_data_block(char *lba_block_data)
{
int count=0;
printf("CMD56 data block dumping:");

while( count < SD_BLOCK_SIZE) {
if(count % 16 == 0)
printf("\n%03d:  ", count);
printf("%02x ", lba_block_data[count]);
count++;
}
printf("\n");

return;
}
```

Demo Test Example

To check the availability and arguments of 'ppeu' in mmc-utils, use the '--help' option under CLI terminal as shown here.

```
$> ./mmc ppeu --help
Usage:
        mmc ppeu read <dump ctrl> <device>
                Usage: mmc ppeu read <-d> <device>
                -d      dump data block
```

The CMD56 default (and only) output with execution of 'ppeu read' are key parameters such as TLC/QLC and SLC utilization percentages. Below is a health information example from a used microSD card showing TLC/QLC and SLC utilization percentages of 21% and 2% respectively.

```
$> ./mmc ppeu read /dev/mmcblk1
Fixed Filed Header: 4Dh 45h 42h 55h
Percentage Step Size: 1
TLC/QLC Percentage Utilization: 21%
SLC Percentage Utilization: 2%
```

When option '-d' is enabled, CMD56 outputs the entire 512-byte health information data block, as shown here.

```
$> ./mmc ppeu read -d /dev/mmcblk1
CMD56 data block dumping:
000:  4d 45 42 55 ff ff ff 01 15 02 ff ff ff ff ff ff
016:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
032:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
048:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
064:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
080:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
096:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
112:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
128:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
144:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
160:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
176:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
192:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
208:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
224:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
240:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
256:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
272:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
288:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
304:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
320:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
336:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
352:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
368:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
384:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
400:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
416:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
432:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
448:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
464:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
480:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
```

```
496:  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
Fixed Filed Header: 4Dh 45h 42h 55h
Percentage Step Size: 1
TLC/QLC Percentage Utilization: 21%
SLC Percentage Utilization: 2%
```

Reference

SD Specifications, Part 1, Physical Layer Specification, version 3.01

Revision History

Rev. C – 07/22

- Updated to include support for i400.
- Removed note from Table 2.

Rev. B – 01/20

- Updated to include support for both i200 and i300.

Rev. A – 09/17

- Initial release

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006
208-368-4000, micron.com/support