Technical Note

Migrating from e·MMC Version 4.4 to 4.41 Devices

Introduction

This technical note describes the changes to the JEDEC *e*•MMC specification from version 4.4 to version 4.41. It also discusses the considerations to make when migrating from Micron®*e*•MMC™ ver. 4.4 to ver. 4.41 devices, including comparisons of the following:

- Micron part numbers
- *e*•MMC registers
- Multilevel cell (MLC) to single-level cell (pSLC) setting flow

JEDEC e•MMC Standard Changes

The most significant items added to the JEDEC *e*•MMC standard (i.e., JEDEC Standard No. 84-A441), from ver. 4.4 to ver. 4.41, are as follows:

- Enhanced reliable write: No limit on the size of a reliable write (see the Enhanced Reliable Write section).
- Background operations: Operations that the device executes when it is not servicing the host (see the Background Operations section).
- High-priority interrupt (HPI) mechanism: A mechanism that enables servicing high-priority requests by allowing the device to interrupt a lower-priority operation before it is complete (see the High-Priority Interrupt section).

Additional changes include:

- Added clarification for command operation in RPMB partition.
- Added clarification of address sequence for user area, including enhanced area and error condition for partition configuration.
- Added clarification for error behavior when partition is configured without setting ERASE_GROUP_DEF and clarification for the device behavior in case the device received a WRITE/ERASE command when the condition of ERASE_GROUP_DEF bit has been changed from the previous power cycle.
- Added clarification for C_SIZE and SEC_COUNT after configuring partitions.
- Added clarification for the configuration of boot and alternative boot operation.
- Added clarification for LOCK_UNLOCK feature of the e•MMC.
- HPI background and one possible solution.
- Introduced new extended CSD registers.
- Corrected equation of general purpose partition size and enhanced user data area size.
- Removed File formats for the *e*•MMC section, formerly section 14.

Note:

To view all JEDEC *e*•MMC standard changes from ver. 4.4 to 4.41, refer to the Changes from version 4.4 to 4.41 section of JEDEC Standard No. 84-A441.

Enhanced Reliable Write

A reliable write is a multiple-block write with a predefined block count that includes reliable write parameters. This transaction is similar to a basic, predefined multiple-block write because the host must use the SET_BLOCK_COUNT command CMD23 immediately preceding the WRITE_MULTIPLE_BLOCK command CMD25. A reliable write differs from the predefined multiple-block write as follows:

- A logical address points to the old data, which must remain unchanged until new data is written to the same logical address and is successfully programmed. This ensures that the target address updated by the reliable write transaction never contains undefined data.
- Data must remain valid even if a sudden power loss occurs during programming.

The reliable write has two implementations: legacy (supported by both ver. 4.4 and 4.41 of the JEDEC *e*•MMC standard) and enhanced (supported only by JEDEC *e*•MMC standard ver. 4.41).

The type of reliable write that is supported by a JEDEC *e*•MMC standard ver. 4.41 device is indicated by the EN_REL_WR bit in the WR_REL_PARAM (EXT_CSD [166]). The two cases are shown as follows:

1. EN_REL_WR = 0
    - At a maximum, two sizes of reliable write transactions are supported: 512B and the reliable write sector count (REL_WR_SEC_C) parameter in EXT_CSD [222] multiplied by 512B.
    - The function is activated by setting the reliable write request parameter (bit 31) to 1 in the SET_BLOCK_COUNT command (CMD23) argument. The REL_WR_SEC_C parameter in EXT_CSD indicates the supported write sector count.
    - The RELIABLE WRITE function is only supported under the following conditions:
        – The length of the WRITE operation equals the supported reliable write size or 512B.
        – The start address of the RELIABLE WRITE operation is aligned to the length of the operation (i.e., the RELIABLE WRITE start address is always a multiple of its own length).
        – The reliable write request is active.
    - When the RELIABLE WRITE function is not supported, the transaction is handled as a basic, predefined multiple-block write case. When the length of the WRITE operation is set to 0, the operation is executed as a basic, open-ended multiple-block-write case, even when the reliable write request is active.
2. EN_REL_WR = 1
    - The reliable write size has no limit.
    - The function is activated by setting the reliable write request parameter (bit 31) to 1 in the SET_BLOCK_COUNT command (CMD23) argument.
    - Reliable write transactions must be sector-aligned; if a reliable write is not sector-aligned, the error bit 19 is set and the transaction does not complete.
    - If a power failure occurs during a RELIABLE WRITE operation, each sector modified by the operation is atomic. After the power failure, sectors modified by the interrupted RELIABLE WRITE operation may contain old data or new data.

- When a RELIABLE WRITE operation is interrupted by an HPI operation, the sectors that the register marks as complete contain new data and the remaining sectors contain old data.
- The REL_WR_SEC_C register should be set to 1 and has no impact on the RELIABLE WRITE operation.

Background Operations

Devices must perform several internal maintenance operations. In order to reduce latencies during time-critical operations like READ and WRITE, it is best to execute maintenance operations at other times, like when the host is not being serviced. Operations are separated into two types:

- Foreground operations: Operations that require the device to service the host, such as READ or WRITE commands.
- Background operations: Operations that the device executes when not servicing the host.

Since foreground operations are higher priority than background operations, the host may interrupt ongoing background operations using the HPI mechanism (see the High-Priority Interrupt section).

When the device is not needed by the host, the host writes any value to BKOPS_START (EXT_CSD [164]), which manually starts background operations. Since the device stays busy until the background operations are complete, the host can use any system idle time to allow the device to perform background operations.

The host sets bit [0] of BKOPS_EN (EXT_CSD [163]) to communicate to the device that background operations are going to start periodically. The device can then delay some maintenance operations until the host writes to BKOPS_START.

The device reports background operation status in bits [1:0] of BKOPS_STATUS (EXT_CSD [246]). These bits represent the urgency of the background operations. There are four status levels:

1. 0x0: No operations required
2. 0x1: Operations outstanding – non critical
3. 0x2: Operations outstanding – performance being impacted
4. 0x3: Operations outstanding – critical

The host checks the status periodically and starts background operations as needed. This allows enough time for the device to perform its maintenance operations, which helps to reduce latencies during the foreground operations. If the status is level 4 (i.e., critical), some operations may extend beyond their original timeouts because of maintenance operations that can no longer be delayed. Ideally, the host gives the device enough time for background operations to avoid reaching a critical status in the first place.

To allow the host to quickly detect the higher levels, the bit 6 (URGENT_BKOPS) in the card status is set whenever the status reaches level 4. So, if URGENT_BKOPS is set, the device needs background operations urgently. This allows the hosts to detect urgent levels on every R1-type response. The host still reads the full status from the BKOPS_STATUS byte periodically and starts background operations as needed.

The background operations can start in the following modes:

- Default (automatic) mode: When the URGENT_BKOPS bit is high, the host sends a PROGRAM or ERASE command to the card, and during the busy, starts a refresh.
- Manual mode: When the URGENT_BKOPS bit is high, the host must set the bit (BKOPS_START) with CMD6, so after the read, during the busy of CMD6, the card starts a refresh.

The background operations feature is optional in JEDEC *e*•MMC ver. 4.41; it is supported in Micron *e*•MMC.

High-Priority Interrupt

In some scenarios, different types of data on the device may have different priorities for the host. The high-priority interrupt (HPI) mechanism enables high-priority requests to be serviced by allowing the device to interrupt a lower priority operation before it is actually completed within OUT_OF_INTERRUPT_TIME (EXT_CSD [198]) timeout (order of tens of ms). To complete the original request, the host may need to repeat either part or all of the interrupted operation. The HPI mechanism is enabled issuing CMD12 (STOP_TRANSMISSION) with HPI bit [0] set.

Before starting the HPI mechanism, the host enables the HPI mechanism by setting the HPI_EN bit in HPI_MGMT (EXT_CSD [161]).

An HPI is only executed during a programming state. It indicates to the device that a higher priority command is pending; therefore, it should interrupt the current operation and return to a transfer state with a different timeout value as soon as possible. If CMD12 is used with an HPI bit set, it differs from the non-HPI command in the allowed state transitions. If an HPI is received in a state other than the programming state:

- If the state transition is allowed, response is sent but the HPI bit is ignored.
- If the state transition is not allowed, the command is regarded as an illegal command.

The HPI command is accepted as a legal command in the program state. However, only the commands below may be interrupted by HPI:

- CMD24 (WRITE_BLOCK)
- CMD25 (WRITE_MULTIPLE_BLOCK)
- CMD38 (ERASE)
- CMD6 (SWITCH) – May only be interrupted when writing to the BKOPS_START field in EXT_CSD

If an HPI is received during commands that cannot be interrupted, the HPI command sends a response but it has no effect and the original command is completed normally —possibly exceeding the OUT_OF_INTERRUPT_TIME timeout. If an HPI is supported, REL_WR_SEC_C is always 1.

The HPI feature is optional in JEDEC *e*•MMC ver. 4.41; it is supported in Micron *e*•MMC.

e•MMC ver. 4.4 and ver. 4.41 Features Comparison

The table below compares the Micron part numbers of *e*•MMC ver. 4.4 and ver. 4.41.

Table 1: Micron Part Numbers Features Comparison

Density	e•MMC ver. 4.4		e•MMC ver. 4.41	
	Micron Part Number	Package Dimensions	Micron Part Number	Package Dimensions
4GB	MTFC4GGQDQ-IT	100-ball LBGA, 14x18x1.4	N2M400FDB311A3CE/F	100-ball LBGA, 14x18x1.4
8GB	MTFC8GKQDQ-IT	100-ball LBGA, 14x18x1.4	N2M400GDB321A3CE/F	100-ball LBGA, 14x18x1.4
16GB	MTFC16GKQDQ-IT	100-ball LBGA, 14x18x1.4	N2M400HDB321A3CE/F	100-ball LBGA, 14x18x1.4
4GB	MTFC4GGQDM-IT	153-ball FBGA, 11.5x13x1.2	MTFC4GMVEA-4M IT	153-ball WFBGA, 11.5x13x0.8
4GB	MTFC4GGQDI-IT	169-ball FBGA,12x16x1.2	MTFC4GMVEA-4M IT	153-ball WFBGA, 11.5x13x0.8
8GB	MTFC8GKQDI-IT	169-ball FBGA, 12x16x1.2	MTFC8GLVEA-4M IT	153-ball WFBGA, 11.5x13x0.8
16GB	MTFC16GKQDI-IT	169-ball FBGA, 12x16x1.2	MTFC16GJVEC-4M IT	169-ball WFBGA, 14x18x0.8
32GB	MTFC32GKQDH-IT	169-ball FBGA, 12x18x1.4	MTFC32GJVED-4M IT	169-ball VFBGA, 14x18x1.0

Notes:
1. Ver. 4.41 is backward-compatible, which includes ballouts.
2. If your ver. 4.4 device is not listed above, contact your Micron representative for assistance with the Micron part number transition.
3. All dimensions in millimeters (mm).

The tables below show the main differences between ECSD and CSD registers from ver. 4.4 to ver. 4.41.

Table 2: ECSD Register Comparison

Features	Micron ver. 4.4 e•MMC	Micron ver. 4.41 e•MMC	ECSD Byte[1]
High-priority interrupt (HPI)	Not supported	Supported	503
Background operation	Not supported	Supported	502
Double data rate (DDR) during boot	Not supported	Supported	228
Maximum timeout for switch CMD when switching partitions	Not defined	Defined	199
Maximum timeout to close a CMD interrupted by HPI	Not defined	Defined	198
DDR	Not supported	Supported	196
Enhanced reliable write	Not supported	Supported [2]	166.2
Write reliability settings register	Not supported	Supported [2]	167
Enhanced partition feature	1. Different setting flow [3] 2. No maximum enhanced size defined in ECSD register; it is deduced by total sector count	1. Different setting flow [3] 2. Maximum enhanced size defined in ECSD register	159:157

Notes:
1. See the Extended CSD register section of JEDEC Standard No. 84-A441.
2. See the appropriate Micron data sheet to determine if this feature is enabled.
3. See the MLC to pSLC Setting Flow Comparison section of this technical note.

Table 3: CSD Register Comparison

Features	Micron ver. 4.4 e•MMC	Micron ver. 4.41 e•MMC	CSD Bit
CSD structure	0x2: CSD version 1.2	0x3: CSD structure version defined in ECSD[194]	127:126
Drive stage register (DSR)	DSR not implemented	DSR implemented	76

MLC to pSLC Setting Flow Comparison

According to the JEDEC standard, the user area may be split into two different types: MLC and enhanced SLC (pSLC). It is also possible for either MLC or pSLC to take up the entire user area.

In an MLC device, each metal-oxide-semiconductor field-effect transistor (MOSFET) can store 2 bits; in a pSLC device, each MOSFET can store 1 bit. For this reason, pSLC is known to have higher reliability, endurance, and performance compared to MLC. See the related NAND device data sheet for more information.

The table below compares the setting flow used to configure the entire user area of a 4GB MLC-based device to pSLC mode from ver. 4.4 to ver. 4.41.

Table 4: MLC to pSLC Setting Flow Comparison

Flow Step in ver. 4.4[1]	Comments	Difference in ver. 4.41
CMD6(175,0x01);cmd13(,);	ERASE_GROUP_DEF field (byte 175)	CMD13 is not needed
CMD6(139,0x00);cmd13(,);	ENH_START_ADDR field (bytes[139:136])	CMD13 is not needed
CMD6(138,0x00);cmd13(,);	Start address 00000	CMD13 is not needed
CMD6(137,0x00);cmd13(,);	–	CMD13 is not needed
CMD6(136,0x00);cmd13(,);	–	CMD13 is not needed
CMD6(142,0x00);cmd13(,);	ENH_SIZE_MULT field (bytes[142:140])	CMD13 is not needed
CMD6(141,0x00);cmd13(,);	–	CMD13 is not needed
CMD6(140,0x39);cmd13(,);	Multiplier of 0x000039	CMD13 is not needed; instead of hard coding 0x000039 as a multiplier, read out CMD8 to read ECSD value and determine the maximum supported enhanced size
CMD6(156,0x01);cmd13(,);	PARTITIONS_ATTRIBUTE field (byte 156) (enhanced user area)	–
CMD6(155,0x01);cmd13(,);	PARTITION_SETTING_COMPLETED field (byte 155)	–
Power cycle	–	No extra power cycle needed
CMD0	–	Not needed
Pause for 2 or more seconds	–	No delay needed
CMD1 can be used to verify that e•MMC is ready	Ready response = 0xc0ff8080	–
Power cycle	–	–
Normal init sequence	–	–

Note: 1. The following code can be used as reference:

CMD Number (ESCD Byte Address [decimal],Value to Write [hex])

Revision History

Rev. A – 08/12

- Initial release

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900
www.micron.com/productsupport Customer Comment Line: 800-932-4992

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.