CE-ATA Host
Design Guidance

Revision 1.0
29-September-2005

Apple Computer, Inc.

Hitachi Global Storage Technologies, Inc.

Intel Corporation

Marvell Semiconductor, Inc.

Nokia Corporation

Seagate Technology LLC

Toshiba America Information Systems, Inc.

The CE-ATA Host Design Guide is available for download at www.ce-ata.org.

DESIGN GUIDE DISCLAIMER

THIS DESIGN GUIDE IS PROVIDED TO YOU "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE AUTHORS OF THIS DESIGN GUIDE DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, RELATING TO USE OR IMPLEMNETATION OF INFORMATION IN THIS SPECIFICATION. THE AUTHORS DO NOT WARRANT OR REPRESENT THAT SUCH USE WILL NOT INFRINGE SUCH RIGHTS. THE PROVISION OF THIS SPECIFICATION TO YOU DOES NOT PROVIDE YOU WITH ANY LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS.

For more information about CE-ATA, refer to the CE-ATA Workgroup website at www.ce-ata.org.

CE-ATA Workgroup Digital Technical Editor:

Amber Huffman
Intel Corporation
2111 NE 25th Ave M/S JF2-53
Hillsboro, OR 97124 USA
Tel: (503) 264-7929
Email: amber.huffman@intel.com

Table of Contents

1. Introduction

CE-ATA is a hard drive interface that is optimized for handheld embedded applications of storage. CE-ATA is layered on top of the MMC electrical interface using a protocol that utilizes the existing MMC access primitives. The interface electrical and signaling definition is as defined in the MMC reference and the CE-ATA Embedded Cable and Connector specification.

The CE-ATA protocol specification primarily discusses device requirements and the device state machine. This document provides additional informative information that may help in the development of a CE-ATA compliant host implementation, including host state machines that describe behavior that is compatible with CE-ATA devices.

One goal of the CE-ATA specification was to allow some class of existing host devices to use CE-ATA devices with only firmware modifications. This document describes how a host can use a CE-ATA device in a data polling fashion that should work with most MMC host implementations. The CE-ATA interrupt mechanism is also described, including potential early implementations of that interrupt mechanism for host implementations.

This document does not duplicate timing requirements that already exist in the CE-ATA protocol specification. For timing requirements, refer to the timing diagrams in section 3 of the CE-ATA protocol specification.

2. Definitions and conventions

2.1. State diagram conventions

For each function to be completed a state machine approach is used to describe the sequence requirements. Each function is composed of several states to accomplish a set goal. Each state of the set is described by an individual state table. Table 1 below shows the general layout for each of the state tables that comprise the set of states for the function.

Table 1 – State Table Cell Description

State name or identifier	Action list[P | W]		
	Branch condition 0	→	**Next state 0**
	Branch condition 1	→	**Next state 1**

Each state is identified by a state designator and a state name. The state designator is unique among all states in all state diagrams. The state designator consists of a set of letters that are capitalized in the title of the figure containing the state diagram followed by a unique number. The state name is a brief description of the primary action taken during the state, and the same state name may appear in other state diagrams. If the same primary function occurs in other states in the same state diagram, they are designated with a unique letter at the end of the name. Additional actions may be taken while in a state and these actions are described in the state description text.

Each transition is identified by a transition label and a transition condition. The transition label consists of the state designator of the state to which the transition is being made. In some cases, the transition to enter or exit a state diagram may come from or go to a number of state diagrams, depending on the command being executed. In this case, the state designator is labeled xx. The transition condition is a brief description of the event or condition that causes the transition to occur and may include a transition action that is taken when the transition occurs. This action is described fully in the transition description text.

Upon entry to a state, all actions to be executed in that state are executed. If a state is re-entered from itself, all actions to be executed in the state are executed again.

It is assumed that all actions are executed within a state and that transitions from state to state are instantaneous.

2.2. References

This document makes reference to the following specifications:

MMC System Specification v 4.0 available to MMCA members under NDA. The CE-ATA specification builds on the MMC specification. Refer to MMCA for IP terms for MMC material.

MMC Systems Summary Specification v 3.31 available at http://www.mmca.org/tech/MMC-System-Summary-v3.31.pdf

ATA on MMC Specification v 1.0 available to MMCA members under NDA. Refer to MMCA for IP terms for MMC material.

AT Attachment with Packet Interface – 6 (ATA/ATAPI-6) [INCITS 361:2002]. Published ATA/ATAPI specifications available from ANSI at webstore.ansi.org or from Global Engineering.

2.3. Definitions

The terminology used in this specification is intended to be self-sufficient and does not rely on overloaded meanings defined in other specifications. Terms with specific meaning not directly clear from the context are clarified in the following sections.

2.3.1. ATA (AT Attachment)

ATA defines the physical, electrical, transport, and command protocols for the internal attachment of storage devices as defined in the ATA reference.

2.3.2. BSY

BSY corresponds to bit 7 in the ATA Status register. BSY is set to one to indicate that the device is busy. The ATA BSY signal has no relationship to the MMC Busy signal. Refer to the ATA reference for more information on the BSY bit.

2.3.3. CE

CE is the acronym used for “Consumer Electronics” and commonly refers to consumer and handheld electronic devices.

2.3.4. CE-ATA sector size

CE-ATA sector size corresponds to the value reported in IDENTIFY DEVICE word 106, refer to Section 4.2.1.4 of CE-ATA Protocol Specification revision 1.0.

2.3.5. Data unit

The term “data unit” describes 512 bytes of data. All CE-ATA data transfers are an integral multiple of data units.

2.3.6. DATx

DATx refers to an MMC data line, where ‘x’ signifies a particular data line (0 through 7). An MMC design may support one, four, or eight data lines. See the MMC reference.

2.3.7. DRQ block

The amount of data transferred in a single RW_MULTIPLE_BLOCK (CMD61) command. This corresponds to the amount of data transferred between assertions of the DRQ bit in the ATA Status register by the device. The DRQ block shall be an integral multiple of the CE-ATA sector size for media access commands. The DRQ block shall be the size of the entire data transfer for the ATA command when interrupts are enabled.

2.3.8. Dword

A Dword is thirty-two (32) bits of data. A Dword may be represented as 32 bits, as two adjacent words, or as four adjacent bytes. When shown as bits the least significant bit is bit 0 and most significant bit is bit 31. The most significant bit is shown on the left. When shown as words the least significant word (lower) is word 0 and the most significant (upper) word is word 1. When shown as bytes the least significant byte is byte 0 and the most significant byte is byte 3. A Dword alignment/granularity means that address/count bits 1-0 are zero.

2.3.9. MMC data block

An MMC data block corresponds to a data transfer on the MMC data lines that includes a start bit, the data to transfer, a 16-bit CRC and the end bit. The size of the MMC data block does not include the start bit, CRC, or the end bit.

2.3.9.1. MMC Busy

MMC Busy corresponds to the device asserting MMC data line DAT0 to indicate to the host that the device is not yet ready to receive data on the MMC bus. The MMC Busy signal has no relationship to the ATA BSY signal. Refer to the MMC reference for more information.

2.3.9.2. word

A word is sixteen (16) bits of data. A word may be represented as 16 bits or as two adjacent bytes. When shown as bits the least significant bit is bit 0 and most significant bit is bit 15. The most significant bit is shown on the left. When shown as bytes the least significant byte (lower) byte is byte 0 and the most significant byte (upper) byte is byte 1. The definition of a word in CE-ATA is the same as the definition of a word in ATA. A word alignment/granularity means that address/count bit 0 is zero.

3. Device Discovery and Initialization

To discover and initialize a CE-ATA device, the host follows a three step process:

- The host determines whether an MMC device is present and performs normal MMC initialization procedures.
- The host checks for the CE-ATA signature using RW_MULTIPLE_BLOCK (CMD60). If the device responds to the RW_MULTIPLE_BLOCK (CMD60) with the CE-ATA signature, a CE-ATA device has been found.
- The host determines whether the MMC device supports ATA mode. If the MMC device supports ATA mode, then the host should select ATA mode.

Detection and initialization of the base MMC device should proceed as defined in the MMC reference.

3.1. Checking for CE-ATA Signature

The CE-ATA signature is comprised of the values placed in the taskfile after power-on, hard reset with GO_IDLE_STATE (CMD0), and software reset. The signature is shown in Figure 1.

The critical values to check for in the CE-ATA signature is the value in the LBA Mid and LBA High registers. If LBA Mid contains CEh and LBA High contains AAh, then the device is a CE-ATA device.

The signature should be read with RW_MULTIPLE_REGISTER (CMD60). If the device is not a CE-ATA device, then no response will be received and the host should move to the next step of determining whether ATA mode is supported. If the device is a CE-ATA device, the device will correctly respond to the RW_MULTIPLE_REGISTER (CMD60) command with the taskfile register contents.

3.2. Selecting ATA Mode

ATA mode allows the device to use FAST_IO (CMD39) in the MMC TRAN state which is a requirement for CE-ATA. Note that CE-ATA 1.0 devices may not show support for ATA mode, even though the device does support FAST_IO (CMD39) in the MMC TRAN state. ATA mode was defined after CE-ATA 1.0 was published.

ATA mode support is indicated in byte 504 (S_CMD_SET) of the EXT_CSD register. If bit 4 is set to ‘1’, then ATA mode is supported. The EXT_CSD register should be read by the host using SEND_EXT_CSD (CMD8). The device will return 512 bytes of data to the host. The host should then check bit 4 of byte 504 in the returned data.

If ATA mode is supported, then the host should select ATA mode. The current mode selected is shown in byte 191 (CMD_SET) of the EXT_CSD register. To select ATA mode, the host should issue the SWITCH (CMD6) command with the appropriate parameters to select ATA mode as the current command set.

Register Address	ATA Register (8-bit)	Reset Value (read)							
0	Reserved	Reserved							
1	Features (exp)	Reserved							
2	Sector Count (exp)	Reserved							
3	LBA Low (exp)	Reserved							
4	LBA Mid (exp)	Reserved							
5	LBA High (exp)	Reserved							
6	Control	Reserved					0 SRST	1 nIEN	0
7	Reserved	Reserved							

8	Reserved	Reserved							
9	Error	Reserved							
10	Sector Count	Reserved							
11	LBA Low	Reserved							
12	LBA Mid	CEh							
13	LBA High	AAh							
14	Device/Head	Reserved						FIO	NBR
15	Status	0 BSY	1 DRDY	R cs	R cs	0 DRQ	R cs	R cs	0 ERR

Figure 1 Device reset signature (initial task file contents)

4. Status and Control Registers

CE-ATA devices contain a set of Status and Control registers that begin at register offset 80h. These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are Dword in size and are Dword aligned. RW_MULTIPLE_REGISTER (CMD60) shall be used to read and write these registers. Note that FAST_IO (CMD39) cannot be used to access these registers since these registers are beyond the address range for FAST_IO (CMD39) and the registers have to be accessed in Dword granularity.

There are several optional registers that may be used to determine the health of the device and whether it has undergone any extreme conditions. These include the temperature (minimum, maximum, and current) registers, the reallocations register, and the retracts register. The current temperature may be used to determine if it is safe to currently spin up the device.

The mandatory capability and control registers are used to determine support for and then control device capabilities. The primary capability that can be changed in CE-ATA 1.0 and 1.1 devices is the MMC data block size.

Status and Control registers may be virtual registers that are not physically implemented on the devices. Hosts should be aware that MMC Busy may be asserted extensively for Status and Control register writes.

5. ATA Data-In Command Protocol

An ATA Data-In command may be executed with interrupts enabled or disabled. Interrupts are enabled by clearing the nIEN bit in the ATA Control register to zero. An example of an ATA Data-In command is READ DMA EXT.

5.1. Interrupts Enabled

The ATA Data-In command protocol when interrupts are enabled is detailed in this section.

The host issues the ATA Data-In command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to zero for the interrupt enabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each data line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

If RW_MULTIPLE_REGISTER (CMD60) was successful, the next step is for the host to issue RW_MULTIPLE_BLOCK (CMD61) to the device. Note that it is illegal for the host to issue a FAST_IO (CMD39) to the device between RW_MULTIPLE_REGISTER (CMD60) and RW_MULTIPLE_BLOCK (CMD61) when interrupts are enabled.

The host issues RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command. The Data Unit Count (specified in 512 byte size units) shall be set to the data transfer size of the entire ATA command; only one RW_MULTIPLE_BLOCK (CMD61) may be used to transfer all of the data for the ATA command. The WR bit shall be set to zero to cause a data transfer from the device to the host.

The host then waits for the device to send an R1 response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1 response.

After receiving the R1 response, the host waits for the device to begin the data transfer. The device will send the data in distinct MMC data blocks. Each MMC data block may be 512 bytes, 1KB, or 4KB depending on the MMC data block size the host negotiated previously with the device. Each distinct MMC data block includes a CRC16 that the host shall use to determine if the data was successfully received. If any of the CRC16 calculations for the MMC data blocks

transferred is invalid, the host shall complete the ATA command with error. No CRC Status is transferred in the case of data transfer from device to host.
If a CRC for a particular MMC data block is invalid and the host desires to abort the ATA command, the host is required to issue the command completion signal disable followed by the STOP_TRANSMISSION (CMD12) command.

When the command is complete, the device will send the command completion signal. This is the device's mechanism to interrupt the host to indicate that the command is complete.

After receiving the command completion signal, the host issues a FAST_IO (CMD39) to the device to determine the ending status of the ATA command. The Register Address should be set to 0Fh, correspond to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

5.2. Polling (Interrupts Disabled)

The ATA Data-In command protocol when the host uses polling (interrupts are disabled) is detailed in this section.

The host issues the ATA Data-In command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to one for the interrupt disabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each data line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

The host then repeatedly issues a FAST_IO (CMD39) to the device to determine the status of the device and readiness to transfer data. The Register Address should be set to 0Fh, corresponding to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. The ATA Status register is repeatedly read until BSY is cleared to zero and DRQ is set to one. The rate for polling the ATA Status register should be chosen to balance power and performance; e.g. a faster polling rate results in higher performance but also consumes more power. The polling rate is design specific.

The host issues RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command. The Data Unit Count (specified in 512 byte size units) shall be set to the amount of data to be transferred between each polling interval, referred to as the DRQ block size. A number of RW_MULTIPLE_BLOCK (CMD61) commands may be used to transfer all of the data for the

ATA command, however the Data Unit Count specified shall correspond to a transfer that is a multiple of the CE-ATA sector size.. The WR bit shall be set to zero to cause a data transfer from the device to the host.

The host then waits for the device to send an R1 response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1 response.

After receiving the R1 response, the host waits for the device to transfer the data. The device will send the data in distinct MMC data blocks. Each MMC data block may be 512 bytes, 1KB, or 4KB depending on the MMC data block size the host negotiated previously with the device. Each distinct MMC data block includes a CRC16 that the host shall use to determine if the data was successfully received. If any of the CRC16 calculations for the MMC data blocks transferred is invalid, the host shall complete the ATA command with error. No CRC Status is transferred in the case of data transfer from device to host.

If a CRC for a particular MMC data block is invalid and the host desires to abort the ATA command, the host is required to issue the STOP_TRANSMISSION (CMD12) command.

When the amount of data requested for the RW_MULTIPLE_BLOCK (CMD61) has been received, if additional data blocks are required to complete the ATA command, the sequence repeats with reading the Status register until DRQ is again set and the next data block is transferred.

After the entire data transfer for the ATA command has been completed, the host issues a FAST_IO (CMD39) to the device to determine the ending status of the command that just completed. The Register Address should be set to 0Fh, corresponding to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. The ATA Status register is read repeatedly until the BSY and DRQ bits are both cleared to zero. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

6. ATA Data-Out Command Protocol

An ATA Data-Out command may be executed with interrupts enabled or disabled. Interrupts are enabled by clearing the nIEN bit in the ATA Control register to zero. An example of an ATA Data-Out command is WRITE DMA EXT.

6.1. Interrupts Enabled

The ATA Data-Out command protocol when interrupts are enabled is detailed in this section.

The host issues the ATA Data-Out command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to zero for the interrupt enabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each DATx line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

If RW_MULTIPLE_REGISTER (CMD60) was successful, the next step is for the host to issue RW_MULTIPLE_BLOCK (CMD61) to the device. Note that it is illegal for the host to issue a FAST_IO (CMD39) to the device between RW_MULTIPLE_REGISTER (CMD60) and RW_MULTIPLE_BLOCK (CMD61) when interrupts are enabled.

The host issues RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command. The Data Unit Count (specified in 512 byte size units) shall be set to the data transfer size of the entire ATA command; only one RW_MULTIPLE_BLOCK (CMD61) may be used to transfer all of the data for the ATA command. The WR bit shall be set to one to cause a data transfer from the host to the device.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be de-asserted. After the MMC Busy signal is de-asserted, indicating the device is ready to receive data, the host then sends the data to the device in distinct MMC data blocks. Each MMC data block may be 512 bytes, 1KB, or 4KB depending on the MMC data block size the host negotiated previously with the device. A CRC16 is inserted on each DATx line by the host following the data transmission. The device may assert MMC Busy between each MMC data block in order to flow

control data from the host. The host shall only issue the next data block to the device when MMC Busy has been de-asserted.

The host receives the CRC Status for each MMC data block immediately following the CRC16 for that block. If the CRC Status is 010b, the transfer of that MMC data block was successful. If the CRC Status is not 010b, the transfer was not successful and the ATA command has failed. If the CRC is invalid for an MMC data block, the host may choose to abort the ATA command. To abort the command, the host is required to issue the command completion signal disable followed by the STOP_TRANSMISSION (CMD12) command; the host is not required to continue data transmission.

When the command is complete, the device will send the command completion signal. This is the device's mechanism to interrupt the host to indicate that the command is complete.

After receiving the command completion signal, the host issues a FAST_IO (CMD39) to the device to determine the ending status of the ATA command. The Register Address should be set to 0Fh, correspond to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

6.2. Polling (Interrupts Disabled)

The ATA Data-Out command protocol when the host uses polling (interrupts are disabled) is detailed in this section.

The host issues the ATA Data-Out command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be set to one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to one for the interrupt disabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each data line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

The host then repeatedly issues a FAST_IO (CMD39) to the device to determine the status of the device and readiness to accept data. The Register Address should be set to 0Fh, corresponding to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. The ATA Status register is repeatedly read until BSY is cleared to zero and DRQ is set to one. The rate for polling the ATA Status register should be chosen to balance power and performance; e.g. a faster polling rate results in higher performance but also consumes more power. The polling rate is design specific.

The host issues RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command. The Data Unit Count (specified in 512 byte size units) shall be set to the amount of data to be transferred between each polling interval, referred to as the DRQ block size.. A number of RW_MULTIPLE_BLOCK (CMD61) commands may be used to transfer all of the data for the ATA command, however the Data Unit Count specified shall correspond to a transfer that is a multiple of the CE-ATA sector size. The WR bit shall be set to one to cause a data transfer from the host to the device.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be de-asserted. After the MMC Busy signal is de-asserted, indicating the device is ready to receive data, the host then sends the data to the device in distinct MMC data blocks. Each MMC data block may be 512 bytes, 1KB, or 4KB depending on the MMC data block size the host negotiated previously with the device. A CRC16 is inserted on each DATx line by the host following the data transmission. The device may assert MMC Busy between each MMC data block in order to flow control data from the host. The host shall only issue the next data block to the device when MMC Busy has been de-asserted.

The host receives the CRC Status for each MMC data block immediately following the CRC16 for that block. If the CRC Status is 010b, the transfer of that MMC data block was successful. If the CRC Status is not 010b, the transfer was not successful and the ATA command has failed. If the CRC is invalid for an MMC data block, the host may choose to abort the ATA command. To abort the command, the host is required to issue the STOP_TRANSMISSION (CMD12) command; the host is not required to continue data transmission.

When the amount of data requested for the RW_MULTIPLE_BLOCK (CMD61) has been transmitted, if additional data blocks are required to complete the ATA command, the sequence repeats with reading the Status register until DRQ is again set to one and the next data block is transferred.

After the entire data transfer for the ATA command has been completed, the host issues a FAST_IO (CMD39) to the device to determine the ending status of the command that just completed. The Register Address should be set to 0Fh, corresponding to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. The ATA Status register is read repeatedly until the BSY and DRQ bits are both cleared to zero. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

7. ATA Non-Data Command Protocol

An ATA Non-Data command may be executed with interrupts enabled or disabled. Interrupts are enabled by clearing the nIEN bit in the ATA Control register to zero. An example of an ATA Non-Data command is STANDBY IMMEDIATE.

7.1. Interrupts Enabled

The ATA Non-Data command protocol when interrupts are enabled is detailed in this section.

The host issues the ATA Non-Data command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to zero for the interrupt enabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each DATx line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

If RW_MULTIPLE_REGISTER (CMD60) was successful, the next step is for the host to issue RW_MULTIPLE_BLOCK (CMD61) to the device to enable the device to send an interrupt for command completion. The Data Unit Count shall be set to 0h to reflect that there is no data transfer. The WR bit shall be set to one in order to allow the device to indicate MMC Busy to the host.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the device to send the command completion signal. This is the device's mechanism to interrupt the host to indicate that the command is complete.

After receiving the command completion signal, the host issues a FAST_IO (CMD39) to the device to determine the ending status of the ATA command. The Register Address should be set to 0Fh, correspond to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

7.2. Polling (Interrupts Disabled)

The ATA Non-Data command protocol when the host uses polling (interrupts are disabled) is detailed in this section.

The host issues the ATA Non-Data command by using RW_MULTIPLE_REGISTER (CMD60) to deliver the ATA command. The Address should be 00h, the Byte Count should be 10h, and the WR bit should be set to one. This corresponds to writing the entire ATA taskfile. The nIEN bit in the ATA Control register is set to one for the interrupt disabled case.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_REGISTER (CMD60). If a response is not received within N_{CR} cycles, an error has occurred and the MMC command will need to be re-issued. CE-ATA devices only include successful status indication in the R1(b) response.

After receiving the R1(b) response, the host waits for the MMC Busy signal to be cleared by the device. When this is cleared, it indicates that the device is prepared to accept the data for the RW_MULTIPLE_REGISTER (CMD60). After MMC Busy is cleared, the host transmits the 16 bytes of data to be written to the taskfile to the device. A CRC16 is inserted on each data line following the data transmission.

The host then receives the CRC Status for the data transfer from the device. If the CRC Status is 010b, the transfer was successful and the host may now consider the command to have been successfully issued. If the CRC Status is not 010b, the transfer was not successful and the RW_MULTIPLE_REGISTER (CMD60) command will need to be re-issued.

If RW_MULTIPLE_REGISTER (CMD60) was successful, the next step is for the host to issue RW_MULTIPLE_BLOCK (CMD61) to the device. The Data Unit Count shall be set to 0h to reflect that there is no data transfer. The WR bit shall be set to one in order to allow the device to indicate MMC Busy to the host.

The host then waits for the device to send an R1(b) response for the RW_MULTIPLE_BLOCK (CMD61). If a response is not received within N_{CR} cycles, an error has occurred and the entire ATA command will need to be re-issued – starting with the RW_MULTIPLE_REGISTER (CMD60) command. Prior to re-issuing a command that has failed, it is recommended that the host first issue a STOP_TRANSMISSION (CMD12) command to ensure the drive and MMC link are in a consistent state. CE-ATA devices shall only include successful status indication in the R1(b) response.

After receiving the R1(b) response and MMC Busy is de-asserted by the device, the host then issues a FAST_IO (CMD39) to the device to determine the ending status of the command that just completed. The Register Address should be set to 0Fh, corresponding to the ATA Status register, and the Register Write flag should be cleared to zero. The host will then receive an R4 response that includes the value of the ATA Status register. The ATA Status register is read repeatedly until the BSY and DRQ bits are both cleared to zero. If an error has occurred, i.e. the ERR bit is set to one in the ATA Status register, the host may want to read additional ATA taskfile registers to determine the nature of the error.

8. Host state machine

8.1. Host MMC State Machine

The MMC state machine describes the MMC behavior for CE-ATA hosts. The MMC layer is decomposed into a command state machine and a data state machine. The command state machine is responsible for the CMD line on the MMC bus and is in control of the MMC layer. The data state machine is responsible for the DATx lines on the MMC bus. The data state machine performs operations as requested by the command state machine and primarily acts as a data movement engine.

HC1: HC_Reset[1]	Reset all host state.		
	1. Internal reset complete	→	HC_ResetDevice
	2. Internal reset not complete	→	HC_Reset
	NOTE: 1. This state is entered asynchronously when the ATA layer requests a hard reset or on power-up.		

HC2: HC_ResetDevice	Transmit GO_IDLE_STATE (CMD0) and complete negotiation to the MMC TRAN state as specified in the MMC reference. Complete all MMC layer initialization, including bus width.		
	1. Host is in MMC TRAN state and MMC layer initialization complete	→	HC_Idle
	2. Host is not in MMC TRAN state or MMC layer initialization not complete	→	HC_ResetDevice

HC3: HC_Idle	Wait for an ATA layer request.		
	1. ATA layer requests STOP_TRANSMISSION (CMD12) command be sent to device and MMC Busy is not asserted	→	HC_Cmd12_Entry
	2. ATA layer requests FAST_IO (CMD39) command be sent to device and MMC Busy is not asserted	→	HC_Cmd39_Entry
	3. ATA layer requests RW_MULTIPLE_REGISTER (CMD60) command be sent to device and MMC Busy is not asserted	→	HC_Cmd60_Entry
	4. ATA layer requests RW_MULTIPLE_BLOCK (CMD61) command be sent to device and MMC Busy is not asserted	→	HC_Cmd61_Entry
	5. No request received from ATA layer or MMC Busy is asserted	→	HC_Idle

HC4: HC_IntWait	Wait for the command completion signal to be received from the device.	
1. ATA layer has not requested command completion signal disable transmission and '0' detected on CMD line	→	HC_IntNotify
2. ATA layer requests command completion signal disable transmission[1]	→	HC_IntCancel
3. ATA layer has not requested command completion signal disable transmission and '0' not detected on CMD line	→	HC_IntWait
NOTE: 1. It is permissible to take transition 1 when the ATA layer has requested command completion signal disable transmission and a '0' has been detected on the CMD line.		

HC5: HC_IntNotify	Notify MMC Data layer to stop any ongoing transmission. Notify ATA layer that the command completion signal has been received.	
1. Unconditional	→	HC_Idle

HC6: HC_IntCancel	Transmit the command completion signal disable to the device (transmit >= four '0's followed by >= one '1').	
1. Command completion signal disable transmission complete	→	HC_Idle
2. Command completion signal disable transmission not complete	→	HC_IntCancel

8.1.1. STOP_TRANSMISSION (CMD12)

HC7: HC_Cmd12_Entry	Notify MMC Data layer to stop any ongoing transmission. Transmit STOP_TRANSMISSION (CMD12) as requested by the ATA layer.	
1. Unconditional	→	HC_Cmd12_R1

HC8: HC_Cmd12_R1	Receive response for STOP_TRANSMISSION (CMD12).	
1. R1(b) response received with valid CRC	→	HC_Cmd12_Notify
2. R1(b) response received with invalid CRC	→	HC_Cmd12_Entry
3. No R1(b) response received and < N_{CR} cycles have elapsed since entry into HC_Cmd12_R1	→	HC_Cmd12_R1
4. No R1(b) response received and >= N_{CR} cycles have elapsed since entry into HC_Cmd12_R1	→	HC_Cmd12_Entry
NOTE: 1. MMC Busy may be asserted on the response for STOP_TRANSMISSION (CMD12) in order to allow the device to flush any volatile buffers to permanent media.		

HC9: HC_Cmd12_Notify	Notify ATA layer that STOP_TRANSMISSION (CMD12) completed successfully.	
1. Unconditional	→	HC_Idle

8.1.2. FAST_IO (CMD39)

HC10: HC_Cmd39_Entry	Transmit FAST_IO (CMD39) with Register Address, Register Data, and Register Write fields as requested by the ATA layer.	
1. Unconditional	→	HC_Cmd39_R4

HC11: HC_Cmd39_R4	Receive response for FAST_IO (CMD39).	
1. R4 response received with valid CRC and status = 1	→	HC_Cmd39_Notify
2. R4 response received with invalid CRC or status = 0	→	HC_Idle[1]
3. No R4 response received and < N_{CR} cycles have elapsed since entry into HC_Cmd39_R4	→	HC_Cmd39_R4
4. No R4 response received and >= N_{CR} cycles have elapsed since entry into HC_Cmd39_R4	→	HC_Idle[1]
NOTE: 1. ATA layer is notified that FAST_IO (CMD39) failed.		

HC12: HC_Cmd39_Notify	Notify ATA layer that FAST_IO (CMD39) completed successfully and deliver Register Data value if the transaction was a register read.	
1. Unconditional	→	HC_Idle

8.1.3. RW_MULTIPLE_REGISTER (CMD60)

HC13: HC_Cmd60_Entry	Transmit RW_MULTIPLE_REGISTER (CMD60) with Address, Byte Count, and WR fields as requested by the ATA layer.	
1. Unconditional	→	HC_Cmd60_R1

HC14: HC_Cmd60_R1	Receive response for RW_MULTIPLE_REGISTER (CMD60).	
1. R1(b) response received with valid CRC	→	HC_Cmd60_Data
2. R1(b) response received with invalid CRC	→	HC_Idle[1]
3. No R1(b) response received and < N_{CR} cycles have elapsed since entry into HC_Cmd60_R1	→	HC_Cmd60_R1
4. No R1(b) response received and >= N_{CR} cycles have elapsed since entry into HC_Cmd60_R1	→	HC_Idle[1]
NOTE: 1. ATA layer is notified that RW_MULTIPLE_REGISTER (CMD60) failed.		

HC15: HC_Cmd60_Data	Notify MMC Data layer that data may be transferred.	
1. Unconditional	→	HC_Idle

8.1.4. RW_MULTIPLE_BLOCK (CMD61)

HC16: HC_Cmd61_Entry	Transmit RW_MULTIPLE_BLOCK (CMD61) with Data Unit Count and WR fields as requested by the ATA layer.	
1. Unconditional	→	HC_Cmd61_R1

HC17: HC_Cmd61_R1	Receive response for RW_MULTIPLE_BLOCK (CMD61).	
1. R1(b) response received with valid CRC	→	HC_Cmd61_Data
2. R1(b) response received with invalid CRC	→	HC_Idle[1]
3. No R1(b) response received and < N_{CR} cycles have elapsed since entry into HC_Cmd61_R1	→	HC_Cmd61_R1
4. No R1(b) response received and >= N_{CR} cycles have elapsed since entry into HC_Cmd61_R1	→	HC_Idle[1]
NOTE: 1. ATA layer is notified that RW_MULTIPLE_BLOCK (CMD61) failed.		

HC18: HC_Cmd61_Data	Notify MMC Data layer that data may be transferred.	
1. ATA layer has indicated interrupts are enabled	→	HC_IntWait
2. ATA layer has indicated interrupts are disabled	→	HC_Idle

8.2. MMC Data State Machine

The MMC block size for all transfers with RW_MULTIPLE_BLOCK (CMD61) shall be 512 bytes, 1KB, or 4KB. The host selects the MMC block size by setting bits 1:0 appropriately in the scrControl register. The MMC block size selected by the host must be supported by the device, as indicated in bits 2:0 of the scrCapabilities register. The MMC block size shall be set to 512 bytes when the ATA command being completed is IDENTIFY DEVICE.

HD1: HD_Idle	Wait for MMC Command layer instruction.	
1. MMC Command layer has indicated data may be transferred and MMC Busy is de-asserted	→	HD_XferType
2. MMC Command layer has not indicated data may be transferred or MMC Busy is asserted	→	HD_Idle

HD2: HD_XferType	Decode MMC transfer type.	
1. MMC command was RW_MULTIPLE_REGISTER (CMD60) with WR=0 (R)	→	HD_Cmd60R_Entry
2. MMC command was RW_MULTIPLE_REGISTER (CMD60) with WR=1 (W)	→	HD_Cmd60W_Entry
3. MMC command was RW_MULTIPLE_BLOCK (CMD61) with WR=0 (R)	→	HD_Cmd61R_Entry
4. MMC command was RW_MULTIPLE_BLOCK (CMD61) with WR=1 (W)	→	HD_Cmd61W_Entry

8.2.1.1.1. RW_MULTIPLE_REGISTER (CMD60) Read Data States

HD3: HD_Cmd60R_Entry	Receive requested register contents from the device.		
	1. MMC Command layer requested data transfer stop	→	HD_Idle
	2. Transfer of register contents and CRC complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd60R_ChkCrc
	3. Transfer of register contents and CRC not complete and MMC Command layer has not requested data transfer stop and < N_{ACIO} cycles have elapsed since entry into HD_Cmd60R_Entry	→	HD_Cmd60R_Entry
	4. Transfer of register contents and CRC not complete and MMC Command layer has not requested data transfer stop and >= N_{ACIO} cycles have elapsed since entry into HD_Cmd60R_Entry	→	HD_Cmd60R_Error

HD4: HD_Cmd60R_ChkCrc	Calculate CRC based on data received and compare to received CRC.		
	1. Calculated CRC and received CRC are equal	→	HD_Cmd60R_Success
	2. Calculated CRC and received CRC are different	→	HD_Cmd60R_Error

HD5: HD_Cmd60R_Success	Notify ATA layer that RW_MULTIPLE_REGISTER (CMD60) was successfully completed.		
	1. Unconditional	→	HD_Idle

HD6: HD_Cmd60R_Error	Notify ATA layer that RW_MULTIPLE_REGISTER (CMD60) was completed in error. For safety, ATA layer should issue command completion signal disable prior to issuing more commands if interrupts are enabled.		
	1. Unconditional	→	HD_Idle

8.2.1.1.2. RW_MULTIPLE_REGISTER (CMD60) Write Data States

HD7: HD_Cmd60W_Entry	Transmit register contents to the device.		
	1. MMC Command layer requested data transfer stop	→	HD_Idle
	2. Transfer of register contents and CRC complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd60W_ChkCrc
	3. Transfer of register contents and CRC not complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd60W_Entry

HD8: HD_Cmd60W_ChkCrc	Receive CRC status for the register contents transferred.		
	1. CRC status reception finished and a positive CRC status of 010b is indicated on DAT0	→	HD_Cmd60W_Success
	2. CRC status reception finished and a positive CRC status of 010b is not indicated on DAT0	→	HD_Cmd60W_Error
	3. CRC status reception not finished	→	HD_Cmd60W_ChkCrc

HD9: HD_Cmd60W_Success	Notify ATA layer that RW_MULTIPLE_REGISTER (CMD60) was successfully completed.	
1. Unconditional	→	HD_Idle

HD10: HD_Cmd60W_Error	Notify ATA layer that RW_MULTIPLE_REGISTER (CMD60) was completed in error.	
1. Unconditional	→	HD_Idle

8.2.1.1.3. RW_MULTIPLE_BLOCK (CMD61) Read Data States

HD11: HD_Cmd61R_Entry	Receive MMC data block and CRC from the device.	
1. MMC Command layer requested data transfer stop	→	HD_Idle
2. Reception of MMC data block and CRC complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd61R_ChkCrc
3. Reception of MMC data block and CRC not complete and MMC Command layer has not requested data transfer stop and < N_{ACIO} cycles have elapsed since entry into HD_Cmd61R_Entry	→	HD_Cmd61R_Entry
4. Reception of MMC data block and CRC not complete and MMC Command layer has not requested data transfer stop and >= N_{ACIO} cycles have elapsed since entry into HD_Cmd61R_Entry	→	HD_Cmd61R_Error

HD12: HD_Cmd61R_ChkCrc	Calculate CRC based on data received and compare to received CRC.	
1. Calculated CRC and received CRC are equal	→	HD_Cmd61R_ChkCnt
2. Calculated CRC and received CRC are different	→	HD_Cmd61R_Error

HD13: HD_Cmd61R_ChkCnt	Notify ATA layer that MMC data block was received.	
1. Data transmission satisfying the Data Unit Count specified in RW_MULTIPLE_BLOCK (CMD61) not finished	→	HD_Cmd61R_Entry
2. Data transmission satisfying the Data Unit Count specified in RW_MULTIPLE_BLOCK (CMD61) finished	→	HD_Cmd61R_Success

HD14: HD_Cmd61R_Success	Notify ATA layer that RW_MULTIPLE_BLOCK (CMD61) was successfully completed.	
1. Unconditional	→	HD_Idle

HD15: HD_Cmd61R_Error	Notify ATA layer that RW_MULTIPLE_BLOCK (CMD61) was completed in error. For safety, ATA layer should issue command completion signal disable prior to issuing more commands if interrupts are enabled.	
1. Unconditional	→	HD_Idle

8.2.1.1.4. RW_MULTIPLE_BLOCK (CMD61) Write Data States

HD16: HD_Cmd61W_Entry	Wait for ATA layer to provide an MMC data block to transfer.	
1. MMC Command layer requested data transfer stop	→	HD_Idle
2. ATA layer has provided one MMC data block to transfer and MMC Command layer has not requested data transfer stop	→	HD_Cmd61W_Xmit
3. ATA layer has not provided one MMC data block to transfer and MMC Command layer has not requested data transfer stop	→	HD_Cmd61W_Entry

HD17: HD_Cmd61W_Xmit	Transmit MMC data block and CRC to device.	
1. MMC Command layer requested data transfer stop	→	HD_Idle
2. Transmission of MMC data block and CRC complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd61W_ChkCrc
3. Transmission of MMC data block and CRC not complete and MMC Command layer has not requested data transfer stop	→	HD_Cmd61W_Xmit

HD18: HD_Cmd61W_ChkCrc	Receive CRC status for the MMC data block transferred.	
1. CRC status reception finished and a positive CRC status of 010b is indicated on DAT0	→	HD_Cmd61W_ChkCnt
2. CRC status reception finished and a positive CRC status of 010b is not indicated on DAT0	→	HD_Cmd61W_Error
3. CRC status reception not finished	→	HD_Cmd61W_ChkCrc

HD19: HD_Cmd61W_ChkCnt	Notify ATA layer that MMC data block transfer complete.	
1. Data transmission satisfying the Data Unit Count specified in RW_MULTIPLE_BLOCK (CMD61) not finished	→	HD_Cmd61W_ChkBusy
2. Data transmission satisfying the Data Unit Count specified in RW_MULTIPLE_BLOCK (CMD61) finished	→	HD_Success

HD20:HD_Cmd61W_ChkBusy	Check if MMC Busy is de-asserted.	
1. MMC Busy is de-asserted	→	HD_Cmd61W_Entry
2. MMC Busy is asserted	→	HD_Cmd61W_ChkBusy

HD21: HD_Cmd61W_Success	Notify ATA layer that RW_MULTIPLE_BLOCK (CMD61) was successfully completed.	
1. Unconditional	→	HD_Idle

HD22: HD_Cmd61W_Error	Notify ATA layer that RW_MULTIPLE_BLOCK (CMD61) was completed in error.	
1. Unconditional	→	HD_Idle

8.2.2. Host ATA State Machine Definition

The ATA state machine describes the required host ATA layer behavior for CE-ATA devices.

Upon host power-up or a host request to perform a hard reset, the host ATA layer shall transition to state HA_Reset. For the sake of clarity, this transition has not been duplicated in all of the defined host states.

HA1: HA_Reset[1]	Reset all host state. Request that the MMC layer perform a hard reset.	
1. Internal reset not complete	→	HA_Reset
2. Internal reset complete	→	HA_Idle
NOTE: 1. This state is entered asynchronously when the ATA layer requests a hard reset or on power-up.		

HA2: HA_Idle	Wait for host to request ATA command transmission, software reset, or register access.	
1. Host requests ATA software reset be completed	→	HA_SoftReset
2. Host requests ATA command be completed and host has not requested a software reset	→	HA_ATACmd
3. Host requests a register access be completed	→	HA_RegAccess
4. No current host request	→	HA_Idle

HA3: HA_ATACmd	Wait for host to request ATA command transmission or software reset.	
1. Host requests ATA non-data command be completed and software reset is not requested	→	HA_ND_Cmd
2. Host requests ATA data-in command be completed and software reset is not requested	→	HA_DI_Cmd
3. Host requests ATA data-out command be completed and software reset is not requested	→	HA_DO_Cmd

HA4: HA_ATACmd_Fail	Notify the host that the ATA command requested could not be completed successfully.	
1. Unconditional	→	HA_Idle

8.2.2.1.1. Host ATA Non-Data Command Protocol

The ATA Non-Data command protocol is defined by the following state tables.

HA5: HA_ND_Cmd	Request that MMC layer transmit RW_MULTIPLE_REGISTER (CMD60) to device with Address = 0, Byte Count = 16, WR = 1, and with data contents as specified by host.	
1. Unconditional	→	HA_ND_CmdChk

HA6: HA_ND_CmdChk	Wait for MMC layer to indicate RW_MULTIPLE_REGISTER (CMD60) completion status.	
1. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed successfully	→	HA_ND_Cmd61Issue
2. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_REGISTER (CMD60) completed	→	HA_ND_CmdChk

HA7: HA_ND_Cmd61Issue	Request that MMC layer transmit RW_MULTIPLE_BLOCK (CMD61) to device with Data Unit Count = 0 and WR = 1.	
1. Unconditional	→	HA_ND_Cmd61Chk

HA8: HA_ND_Cmd61Chk	Wait for MMC layer to indicate RW_MULTIPLE_BLOCK (CMD61) completion status.	
1. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) completed successfully	→	HA_IntChk
2. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_BLOCK (CMD61) completed	→	HA_ND_Cmd61Chk

HA9: HA_ND_IntChk	Determine if interrupts are enabled.	
1. Current value of nIEN in ATA Control register is one	→	HA_ND_StatusRead
2. Current value of nIEN in ATA Control register is zero	→	HA_ND_IntWait

HA10: HA_ND_IntWait	Wait for the MMC layer to indicate the command completion signal was received.	
1. MMC layer has indicated command completion signal was received	→	HA_ND_StatusRead
2. MMC layer has not indicated command completion signal was received	→	HA_ND_IntWait

HA11: HA_ND_StatusRead	Request that MMC layer transmit FAST_IO (CMD39) to device with Register Address = Fh, and Register Write = 0.	
1. Unconditional	→	HA_ND_StatusReadChk

HA12: HA_ND_StatusReadChk	Wait for MMC layer to indicate FAST_IO (CMD39) completion status.	
1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_ND_StatusChk
2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_ND_Cmd

HA13: HA_ND_StatusChk	Check ATA Status register value read with FAST_IO (CMD39).	
1. BSY or DRQ set to one in ATA Status register	→	HA_ND_StatusRead[1]
2. BSY and DRQ cleared to zero in ATA Status register	→	HA_ND_Complete
NOTE: 1. The rate for polling the ATA Status register is design specific. It should be chosen to appropriately balance power and performance for the implementation.		

HA14: HA_ND_Complete	Notify host that command is complete and deliver ATA Status register value as final completion status.	
1. Unconditional	→	HA_Idle

8.2.2.1.2. Host ATA Data-In Command Protocol

The ATA Data-In command protocol is defined by the following state tables.

HA15: HA_DI_Cmd	Request that MMC layer execute RW_MULTIPLE_REGISTER (CMD60) to device with Address = 0, Byte Count = 16, WR = 1, and with data contents as specified by host	
1. Unconditional	→	HA_DI_CmdChk

HA16: HA_DI_CmdChk	Check MMC layer RW_MULTIPLE_REGISTER (CMD60) completion status	
1. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed successfully	→	HA_DI_IntChk
2. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_REGISTER (CMD60) completed	→	HA_DI_CmdChk

HA17: HA_DI_IntChk	Determine if interrupts are enabled.	
1. Current value of nIEN in ATA Control register is one	→	HA_DIP_CheckStatus
2. Current value of nIEN in ATA Control register is zero	→	HA_DII_StartData

HA18: HA_DII_StartData	Request that MMC layer transmit RW_MULTIPLE_BLOCK (CMD61) to device with Data Unit Count set to ATA command transfer size divided by 512 and WR = 0.	
1. Unconditional	→	HA_DII_CMD61Chk

HA19: HA_DII_CMD61Chk	Wait for MMC layer to indicate RW_MULTIPLE_BLOCK (CMD61) response		
	1. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with success	→	HA_DII_IntWait
	2. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with error or R1 response timed out	→	HA_ATACmd_Fail
	3. MMC layer has not indicated RW_MULTIPLE_BLOCK (CMD61) response	→	HA_DII_CMD16Chk

HA20: HA_DII_IntWait	Wait for the MMC layer to indicate the command completion signal was received. Receive data from MMC layer.		
	1. MMC layer has indicated command completion signal was received	→	HA_DII_StatusRead
	2. MMC layer has not indicated command completion signal was received	→	HA_DII_IntWait

HA21: HA_DII_StatusRead	Request that MMC layer transmit FAST_IO (CMD39) to device with Register Address = Fh, and Register Write = 0.		
	1. Unconditional	→	HA_DII_StatusReadChk

HA22: HA_DII_StatusReadChk	Wait for MMC layer to indicate FAST_IO (CMD39) completion status.		
	1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_DII_Complete
	2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_ATACmd_Fail
	3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_DII_StatusReadChk

HA23: HA_DII_Complete	Notify host that command is complete and deliver ATA Status register value as final completion status.		
	1. Unconditional	→	HA_Idle

HA24: HA_DIP_CheckStatus	Request that MMC layer transmit FAST_IO (CMD39) to device with Register Address = Fh, and Register Write = 0.		
	1. Unconditional	→	HA_DIP_StatusReadChk

HA25: HA_DIP_StatusReadChk	Wait for MMC layer to indicate FAST_IO (CMD39) completion status.		
	1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_DIP_ChkComplete
	2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_ATACmd_Fail
	3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_DIP_StatusReadChk

HA26: HA_DIP_ChkComplete	Check ATA Status register value	
1. BSY bit set in Status register value	→	HA_DIP_CheckStatus[1]
2. BSY bit cleared and DRQ bit set in Status register value	→	HA_DIP_StartData
3. BSY bit cleared and DRQ bit cleared in Status register value	→	HA_DIP_Complete
NOTE: 1. The rate for polling the ATA Status register is design specific. It should be chosen to appropriately balance power and performance for the implementation.		

HA27: HA_DIP_Complete	Notify host that command is complete and deliver ATA Status register value as final completion status.	
1. Unconditional	→	HA_Idle

HA28: HA_DIP_StartData	Request that MMC layer transmit RW_MULTIPLE_BLOCK (CMD61) to device with Data Unit Count set to polling DRQ block size[1] and WR = 0.	
1. Unconditional	→	HA_DIP_CMD61Chk
NOTE: 1. The DRQ block size shall be a multiple of the reported CE-ATA sector size.		

HA29: HA_DIP_CMD61Chk	Wait for MMC layer to indicate RW_MULTIPLE_BLOCK (CMD61) response	
1. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with success	→	HA_DIP_ReceiveData
2. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with error or R1 response timed out	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_BLOCK (CMD61) response	→	HA_DIP_CMD61Chk

HA30: HA_DIP_ReceiveData	Receive data from MMC layer	
1. Data reception satisfying issued RW_MULTIPLE_BLOCK command complete	→	HA_DIP_CheckStatus
2. Data reception satisfying issued RW_MULTIPLE_BLOCK command not complete	→	HA_DIP_ReceiveData

8.2.2.1.3. Host ATA Data-Out Command Protocol

The ATA Data-Out command protocol is defined by the following state tables.

HA31: HA_DO_Cmd	Request that MMC layer execute RW_MULTIPLE_REGISTER (CMD60) to device with Address = 0, Byte Count = 16, WR = 1, and with data contents as specified by host	
1. Unconditional	→	HA_DO_CmdChk

HA32: HA_DO_CmdChk	Check MMC layer RW_MULTIPLE_REGISTER (CMD60) completion status	
1. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed successfully	→	HA_DO_IntChk
2. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_REGISTER (CMD60) completed	→	HA_DO_CmdChk

HA33: HA_DO_IntChk	Determine if interrupts are enabled.	
1. Current value of nIEN in ATA Control register is one	→	HA_DOP_CheckStatus
2. Current value of nIEN in ATA Control register is zero	→	HA_DOI_StartData

HA34: HA_DOI_StartData	Request that MMC layer transmit RW_MULTIPLE_BLOCK (CMD61) to device with Data Unit Count set to ATA command transfer size divided by 512 and WR = 1.	
1. Unconditional	→	HA_DOI_CMD61Chk

HA35: HA_DOI_CMD61Chk	Wait for MMC layer to indicate RW_MULTIPLE_BLOCK (CMD61) response	
1. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with success	→	HA_DOI_IntWait
2. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with error or R1 response timed out	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_BLOCK (CMD61) response	→	HA_DOI_CMD61Chk

HA36: HA_DOI_IntWait	Wait for the MMC layer to indicate the command completion signal was received. Transmit data to the MMC layer.	
1. MMC layer has indicated command completion signal was received	→	HA_DOI_StatusRead
2. MMC layer has not indicated command completion signal was received	→	HA_DOI_IntWait

HA37: HA_DOI_StatusRead	Request that MMC layer transmit FAST_IO (CMD39) to device with Register Address = Fh, and Register Write = 0.	
1. Unconditional	→	HA_DOI_ StatusReadChk

HA38: HA_DOI_StatusReadChk	Wait for MMC layer to indicate FAST_IO (CMD39) completion status.	
1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_DOI_Complete
2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_DOI_ StatusReadChk

HA39: HA_DOI_Complete	Notify host that command is complete and deliver ATA Status register value and CRC status values as final completion status.	
1. Unconditional	→	HA_Idle

HA40: HA_DOP_CheckStatus	Request that MMC layer transmit FAST_IO (CMD39) to device with Register Address = Fh, and Register Write = 0.	
1. Unconditional	→	HA_DOP_ StatusReadChk

HA41: HA_DOP_StatusReadChk	Wait for MMC layer to indicate FAST_IO (CMD39) completion status.	
1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_DOP_ChkComplete
2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_ATACmd_Fail
3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_DOP_ StatusReadChk

HA42: HA_DOP_ChkComplete	Check ATA Status register value	
1. BSY bit set in Status register value	→	HA_DOP_CheckStatus[1]
2. BSY bit cleared and DRQ bit set in Status register value	→	HA_DOP_StartData
3. BSY bit cleared and DRQ bit cleared in Status register value	→	HA_DOP_Complete
NOTE: 1. The rate for polling the ATA Status register is design specific. It should be chosen to appropriately balance power and performance for the implementation.		

HA43: HA_DOP_Complete	Notify host that command is complete and deliver ATA Status register value and CRC status values as final completion status.	
1. Unconditional	→	HA_Idle

HA44: HA_DOP_StartData	Request that MMC layer transmit RW_MULTIPLE_BLOCK (CMD61) to device with Data Unit Count set to polling DRQ block size[1] and WR = 1.	
1. Unconditional	→	HA_DOP_CMD61Chk
NOTE: 1. The DRQ block size shall be a multiple of the reported CE-ATA sector size.		

HA45: HA_DOP_CMD61Chk	Wait for MMC layer to indicate RW_MULTIPLE_BLOCK (CMD61) response	
1. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with success	→	HA_DOP_SendData
2. MMC layer has indicated RW_MULTIPLE_BLOCK (CMD61) R1 response received with error or R1 response timed out	→	HA_ATACmd_Fail
3. MMC layer has not indicated RW_MULTIPLE_BLOCK (CMD61) response	→	HA_DOP_CMD16Chk

HA46: HA_DOP_SendData	Deliver transmit data to MMC layer	
1. Data transmission satisfying issued RW_MULTIPLE_BLOCK (CMD61) command complete	→	HA_DOP_CheckStatus
2. Data transmission satisfying issued RW_MULTIPLE_BLOCK (CMD61) command not complete	→	HA_DOP_SendData

8.2.2.1.4. Host Register Access

HA47: HA_RegAccess	Request that MMC layer execute RW_MULTIPLE_REGISTER (CMD60) to device with Address, Byte Count, WR fields and data contents as specified by host	
1. Unconditional	→	HA_RegChk

HA48: HA_RegChk	Check MMC layer RW_MULTIPLE_REGISTER (CMD60) completion status	
1. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed successfully	→	HA_RegComplete
2. MMC layer has indicated RW_MULTIPLE_REGISTER (CMD60) completed with error	→	HA_RegFail
3. MMC layer has not indicated RW_MULTIPLE_REGISTER (CMD60) completed	→	HA_RegChk

HA49: HA_RegComplete	Notify host that the register access requested completed successfully.	
1. Unconditional	→	HA_Idle

HA50: HA_RegFail	Notify host that the register access requested failed	
1. Unconditional	→	HA_Idle

8.2.2.1.5. Software Reset

HA51: HA_SoftReset	Request that MMC layer execute FAST_IO (CMD39) to device with Register Address = 6h, Register Write = 1, and Register Data = 06h	
1. Unconditional	→	HA_SR_Comp1

HA52: HA_SR_Comp1	Check MMC layer FAST_IO (CMD39) completion status	
1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_SR_Issue2
2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_SoftReset
3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_SR_Comp1

HA53: HA_SR_Issue2	Request that MMC layer execute FAST_IO (CMD39) to device with Register Address = 6h, Register Write = 1, and Register Data = 02h	
1. Unconditional	→	HA_SR_Comp2

HA54: HA_SR_Comp2	Check MMC layer FAST_IO (CMD39) completion status	
1. MMC layer has indicated FAST_IO (CMD39) completed successfully	→	HA_Idle
2. MMC layer has indicated FAST_IO (CMD39) completed with error	→	HA_SoftReset
3. MMC layer has not indicated FAST_IO (CMD39) completed	→	HA_SR_Comp2

9. Command Completion Signal Handling

This section describes several possible near-term implementation options for supporting use of the command completion signal with host controllers that may not have been enhanced with direct support for this capability. The command completion signal serves as an interrupt to the host at the end of an ATA command (that completes successfully or with error).

The host does not need to use the command completion signal mechanism; the capability is an optimization for efficient operation and utilization of host compute resources. On resets, the command completion signal is disabled since the nIEN bit in the Device Control register is set to one. However, use of the command completion signal has benefits and it may be desirable to provide host support for it in the near term prior to availability of new host controllers that have this feature comprehended from the start. Figure 2 depicts the essence of the command completion signal mechanism.

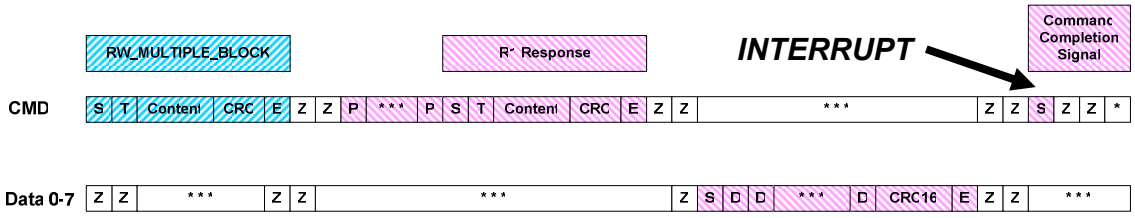


Figure 2 CE-ATA Command Completion Signal Timing Diagram

Note that to use the command completion signal the host implementation shall not actively pull the CMD line high after reception of the response for RW_MULTIPLE_BLOCK (CMD61). After reception of the response for RW_MULTIPLE_BLOCK (CMD61), the host shall tri-state the CMD line in order to make use of the command completion signal. If the host actively pulls the CMD line high after the RW_MULTIPLE_BLOCK (CMD61) response, then the host shall ensure that the nIEN bit in the Device Control register is set to one to disable use of the command completion signal.

9.1. Option 1: Hosts with Multi-Purpose Reconfigurable Ports

Some MMC host controller implementations may have multiple purposes and configurations possible for the pads used to interface to the MMC signals.

For MMC host controller implementations that have a multi-purpose pin used for the MMC CMD line that can also be configured on the fly in software as a GPIO with interrupt on change of state without perturbing the underlying MMC controller state, the CE-ATA interrupt can be implement wholly in firmware by reconfiguring the pin immediately following the R1 response from the RW_MULTIPLE_BLOCK command.

The reconfiguration approach does suffer from a theoretical race condition if the host is slow in performing the reconfiguration, since the interrupt may have occurred prior to the pin reconfiguration being completed. This can be mitigated in practice by recognizing that the smallest CE-ATA media transfer command is 4KB in size, so for a 4-bit wide configuration a minimum of 8000 MMC clocks will elapse after the R1 response before the interrupt signal will fire. For commands that don't transfer data due to device errors in recovering the data, this time will in practice be much longer as disk drives typically perform an exhaustive sequence of retry operations. Commands that fail before being executed due to errors in the command arguments are generally a result of a flaw in the host firmware (such as delivering a command with a block

address past the end of the drive). Such cases of host flaws should be addressed by correcting the source of the problem.

Non-data commands would be best supported with interrupts disabled since they are not as readily bounded and as a result there is not good confidence that the race condition is not encountered. This would be done by setting nIEN bit in the taskfile register to 1 for non-data commands as part of issuing the command.

9.1.1. Block Diagram

Figure 3 illustrates the hardware block diagram for the all-firmware solution possible with the class of hosts that have the capability to reconfigure their ports as described earlier. As the figure indicates, the diagram shows no additional hardware and is indistinguishable from an unmodified configuration.

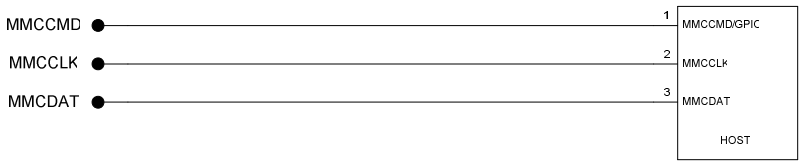


Figure 3 Block diagram for all-firmware solution

9.1.2. Description

Figure 4 outlines the pseudo-code for the basic operations performed for the all-firmware solution with hosts that have reconfigurable ports.

```
Initialize()
{
        ConfigurePort(MMCPORT, MMC_CMD_CONFIG);
}

IssueCMD61(*Args)
{
        *CommandStruct=BuildCommand(CMD61, *Args)
        ControllerInterruptOnR1=TRUE;
        IssueCommand(*CommandStruct);
}

ControllerInterrupt()
{
        ConfigurePort(MMCPORT, GPIO_INPUT_CONFIG);
        ControllerInterruptOnChange=TRUE;
}

GPIOInterrupt()
{
        ConfigurePort(MMCPORT,MMC_CMD_CONFIG);
        RetireCompletedCommand();
}
```

Figure 4 Pseudo-code for all-firmware port reconfiguration approach

9.2. Option 2: Hosts with Available GPIO Ports

Some MMC host controllers may not have the ability to reconfigure the pin used for the CMD line or may produce undesirable side effects due to loss of state in the controller. For controllers that have additional GPIO signals available that support internal interrupt generation on change of state, the MMC CMD signal may be connected to both the host controller's MMC CMD pin as well as an auxiliary GPIO pin used to detect the interrupt signal. This requires that the electrical loading of the additional GPIO connection does not compromise the functionality of the MMC CMD signal.

Because the GPIO interrupt must be enabled at the appropriate time by the firmware, the same race condition considerations as for option #1 still applies.

9.2.1. Block Diagram

Figure 5 outlines the block diagram for a configuration that uses an auxiliary GPIO port on the host.

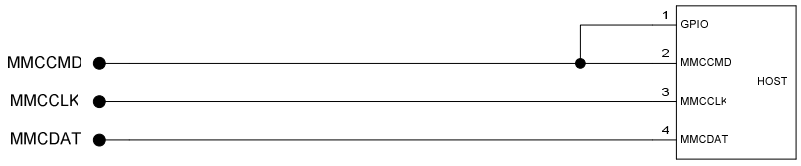


Figure 5 Block diagram of auxiliary GPIO solution

9.2.2. Description

Figure 6 outlines the pseudo-code for the basic operations performed for the firmware solution using an auxiliary GPIO.

```
Initialize()
{
        ConfigurePort(MMCPORT, MMC_CMD_CONFIG);
        ConfigurePort(GPIOPORT,INPUT);
}

IssueCMD61(*Args)
{
        *CommandStruct=BuildCommand(CMD61, *Args)
        ControllerInterruptOnR1=TRUE;
        IssueCommand(*CommandStruct);
}

ControllerInterrupt()
{
        ControllerInterruptOnGPIO=TRUE;
}

GPIOInterrupt()
{
        ControllerInterruptOnGPIO=FALSE;
        RetireCompletedCommand();
}
```

Figure 6 Pseudo-code for auxiliary GPIO approach

9.3. Option 3: External Logic plus GPIO

For controllers that have available GPIO ports but which cannot reliably detect the brief pulse on the CMD line for triggering an interrupt (i.e. level-triggered interrupt) or for which hardware enforcement of the interrupt cancellation signal might be desired, external “glue” logic can be used to provide support for both the interrupt detection and the interrupt cancellation signals.

As before, since firmware is involved in arming the external glue logic in preparation for detecting the interrupt, the same race condition described earlier exists and the same precautions must be taken.

9.3.1. Block Diagram

Figure 7 depicts the external glue logic block diagram.

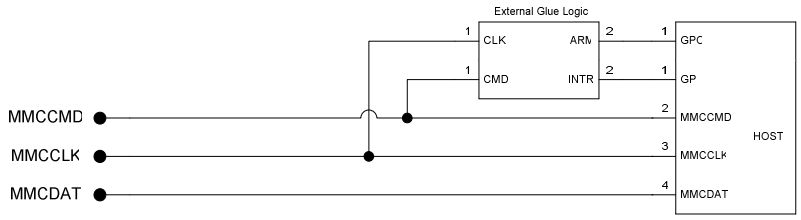


Figure 7 External logic solution

The function of the glue logic is as follows: The glue logic is armed when the ARM signal is asserted by the host GPO. When ARM is asserted, if a transition to zero on the CMD line is detected, the INTR signal is asserted until the ARM signal is de-asserted. If the ARM signal is asserted and then subsequently de-asserted without an interrupt having been received, the glue logic emits the CE-ATA interrupt cancellation signal on the MMC CMD line.

9.3.2. State Machine Definition

The following state tables define the function performed by the external glue logic. The logic is implemented in 8 states which requires three D flip flops plus associated logic gates.

IDLE	Set INTR=0, OE=0, LINE=*		
1. ARM signal asserted		→	ARMED
2. ARM signal not asserted		→	IDLE

ARMED	Set INTR=0, OE=0, LINE=*		
1. ARM signal deasserted		→	ABORTA
2. ARM signal asserted and CMD signal deasserted		→	INTERRUPT
3. ARM signal asserted and CMD signal asserted		→	ARMED

INTERRUPT	Set INTR=1, OE=0, LINE=*		
1. ARM signal deasserted		→	IDLE
2. ARM signal asserted		→	INTERRUPT

ABORTA	Set INTR=0, OE=1, LINE=0		
1. Unconditional		→	ABORTB

ABORTB	Set INTR=0, OE=1, LINE=0		
1. Unconditional		→	ABORTC

ABORTC	Set INTR=0, OE=1, LINE=0		
1. Unconditional		→	ABORTD

ABORTD	Set INTR=0, OE=1, LINE=0		
1. Unconditional		→	ABORTE

ABORTE	Set INTR=0, OE=1, LINE=1		
1. Unconditional		→	IDLE

9.3.3. State Tables and Variables

		Inputs				Outputs		
Current State	State A B C	ARM	CMD	Next State	State A B C	INTR	OE	LINE
IDLE	0 0 0	0	*	IDLE	0 0 0	0	0	*
IDLE	0 0 0	1	*	ARMED	1 0 0	0	0	*
ARMED	1 0 0	0	*	ABORTA	1 0 1	0	0	*

ARMED	1 0 0	1	0	INTERRUPT	1 1 0	0	0	*
ARMED	1 0 0	1	1	ARMED	1 0 0	0	0	*
INTERRUPT	1 1 0	0	*	IDLE	0 0 0	1	0	*
INTERRUPT	1 1 0	1	*	INTERRUPT	1 1 0	1	0	*
ABORTA	1 0 1	*	*	ABORTB	1 1 1	0	1	0
ABORTB	1 1 1	*	*	ABORTC	0 1 1	0	1	0
ABORTC	0 1 1	*	*	ABORTD	0 0 1	0	1	0
ABORTD	0 0 1	*	*	ABORTE	0 1 0	0	1	0
ABORTE	0 1 0	*	*	IDLE	0 0 0	0	1	1

		A/B			
		00	01	11	10
C/ARM	00	0	0	0	1
	01	1	0	1	1
	11	0	0	0	1
	10	0	0	0	1

A' =A B# + A C# ARM + B# C# ARM

		A/B			
		00	01	11	10
C/ARM	00	0	0	0	0
	01	0	0	1	CMD#
	11	1	0	1	1
	10	1	0	1	1

B' = B# C + A C + A B ARM + A B# C# ARM CMD

		A/B			
		00	01	11	10
C/ARM	00	0	0	0	1
	01	0	0	0	0
	11	0	1	1	1
	10	0	1	1	1

C' = B C + A C + A B# ARM#

		A/B			
		00	01	11	10
C	0	0	0	1	0
	1	0	0	0	0

INTR = A B C#

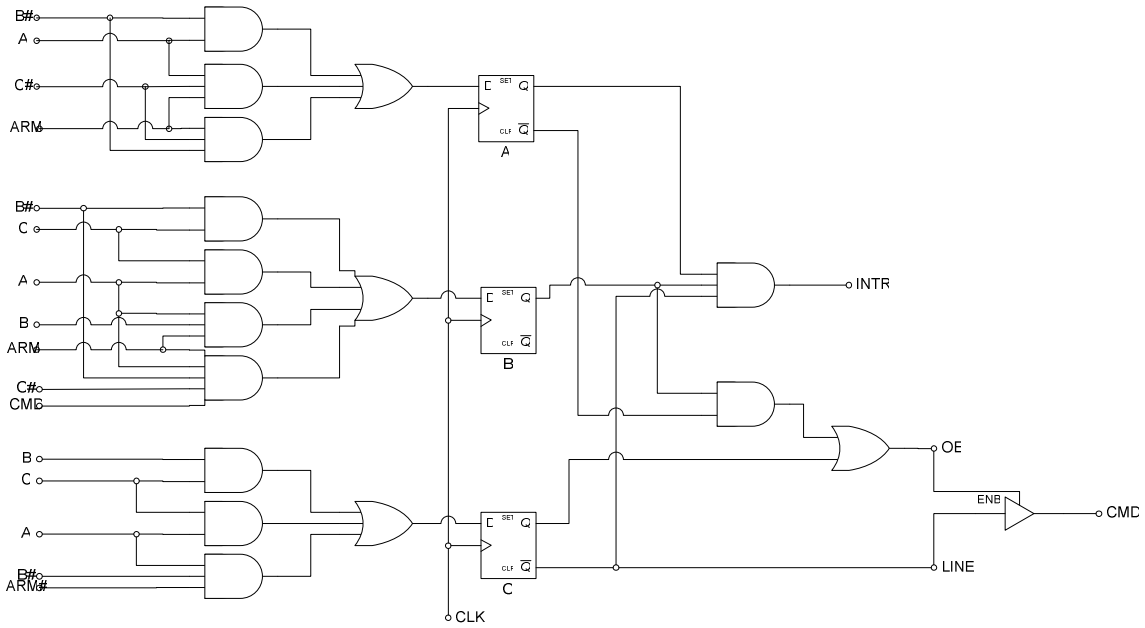
		A/B			
		00	01	11	10
C	0	0	1	0	0
	1	1	1	1	1

OE = C + A# B

		A/B			
		00	01	11	10
C	0	*	1	*	*
	1	0	0	0	0

LINE = C#

9.4. Logic Implementation



10. Error Recovery

Several methods of ATA command failure may occur, including:

- No response to an MMC command, like RW_MULTIPLE_REGISTER (CMD60)
- CRC is invalid for an MMC command or response
- CRC16 is invalid for an MMC data packet
- ATA Status register reflects an error by setting the ERR bit to one
- The command completion signal does not arrive within a host specified time out period

Error conditions are expected to happen infrequently. Thus, a robust error recovery mechanism may be used for each error event. The recommended error recovery procedure after a timeout is:

- Issue the command completion signal disable if nIEN was cleared to zero and the RW_MULTIPLE_BLOCK (CMD61) response has been received
- Issue STOP_TRANSMISSION (CMD12) and successfully receive the R1 response.
- Issue a software reset to the CE-ATA device using FAST_IO (CMD39)

If STOP_TRANMISSION (CMD12) is successful, then the device is again ready for ATA commands. However, if the error recovery procedure does not work as expected or there is another timeout, the next step is to issue GO_IDLE_STATE (CMD0) to the device. GO_IDLE_STATE (CMD0) is a hard reset to the device and completely resets all device state. Note that after issuing GO_IDLE_STATE (CMD0), all device initialization needs to be completed again.

If the CE-ATA device completes all MMC commands correctly but fails the ATA command with the ERR bit set in the ATA Status register, no error recovery action is required. The ATA command itself failed implying that the device could not complete the action requested, however, there was no communication or protocol failure. After the device signals an error by setting the ERR bit to one in the ATA Status register, the host may attempt to retry the command.