**Please note that Cypress is an Infineon Technologies Company.**
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN224414

# Using SDHC Host Controller in Traveo II Family MCU

**Author: Kenichi Sunada**

**Associated Part Family: Traveo™ II Family CYT3/4 Series**

AN224414 describes the data transaction sequence using a secure digital high capacity (SDHC) host controller as part of the Traveo™ II Family MCU.

## Contents



## 1 Introduction

This application note describes the data and abort transaction sequences using a SDHC host controller for Cypress Traveo II family MCUs. The SDHC host controller allows interfacing embedded multimedia card (eMMC) based memory devices, secure digital (SD) cards, and secure digital input output (SDIO) cards. This application note also describes the transaction sequences with and without Direct Memory Access (DMA) functionality. The initialization sequence is described as part of “SDHC Host Controller” chapter of the Architecture Technical Reference Manual (TRM).

This application note is intended for CYT3/4 series of MCUs. The CYT3 series has one Arm® Cortex®-M7-based CPU (CM7) and one Cortex-M0+-based CPU (CM0+), while the CYT4 series has two CM7 CPUs and one CM0+ CPU.

To understand more about the functionality described and the terminologies used in this application note, see the “SDHC Host Controller” chapter of the Architecture TRM.

## 2 Data Transaction Sequence

Depending on whether DMA is used or not, there are two execution methods. The sequence without using DMA transfer is shown in Figure 1; the sequence with the DMA is shown in Figure 2 and Figure 3.

In addition, the sequences of transfers are classified into the following types according to the number of blocks specified:

(1) Single-Block Transfer:

The number of blocks is specified to the host controller before the transfer, and is always one.

(2) Multi-Block Transfer:

The number of blocks is specified to the host controller before the transfer, and shall be one or more.

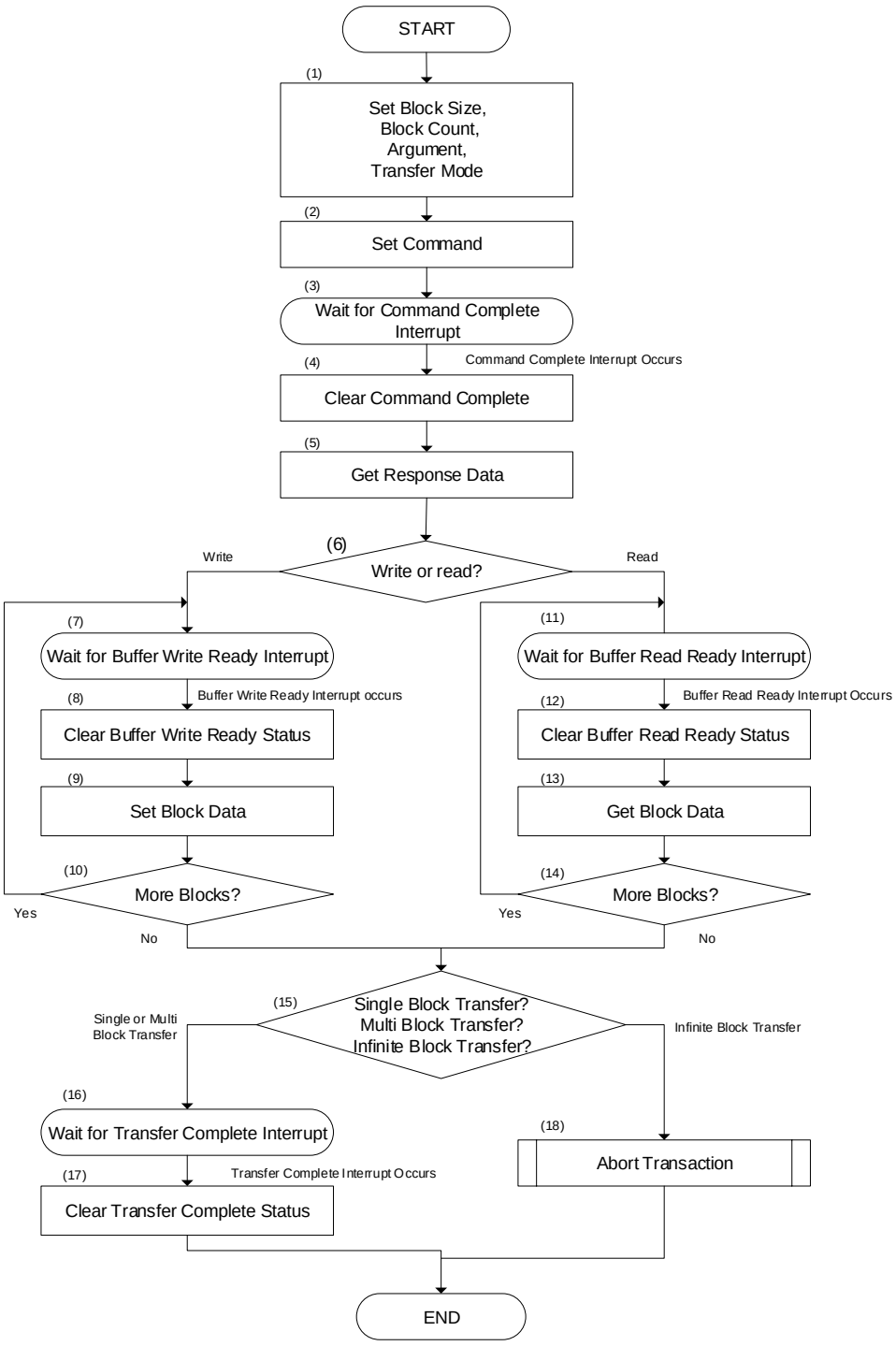
(3) Infinite Block Transfer:

The number of blocks is not specified to the host controller before the transfer. This transfer is continued until an abort transaction is performed. This abort transaction is performed by CMD12 in the case of an SD memory card and eMMC, and by CMD52 in the case of an SDIO card.

## 2.1 Data Transaction Sequence Without DMA

Figure 1 shows the data transaction sequence without DMA.

Figure 1. Data Transaction Sequence without DMA




The points highlighted in the above flowchart are described as follows:

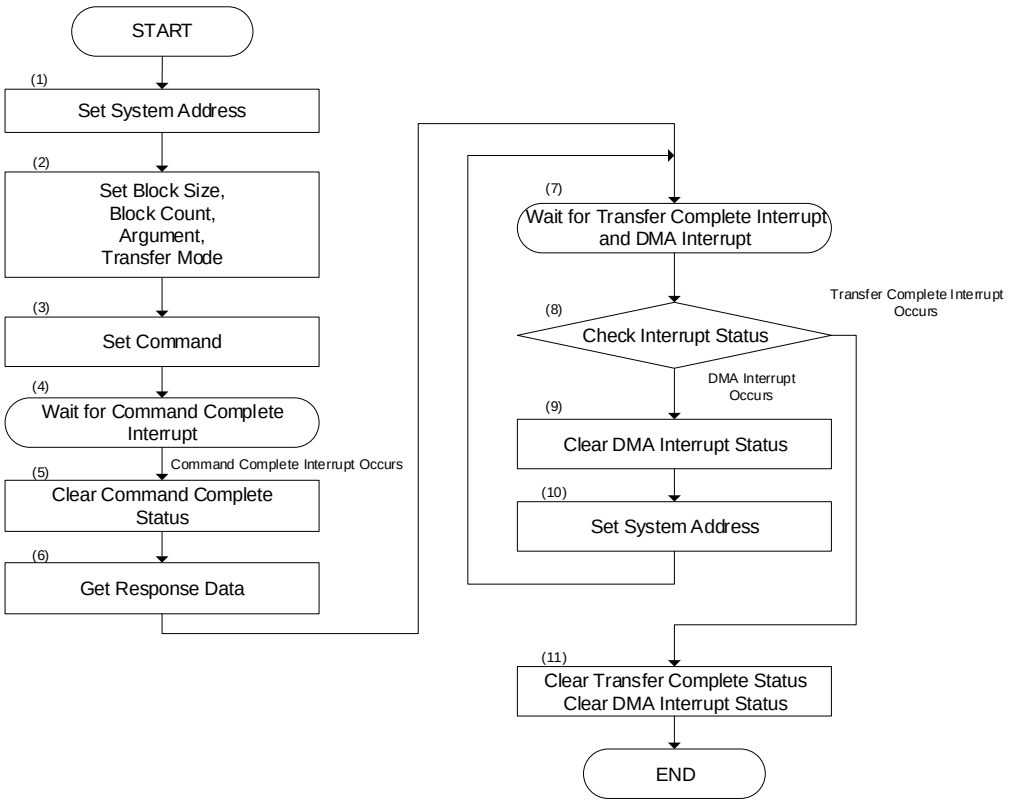
1. Configure the following entities:

   a. Block size register (SDHC_CORE_BLOCKSIZE_R) to the required length of a block.
   b. Block count register (SDHC_CORE_BLOCKCOUNT_R) to the required data block count.
   c. Argument register (SDHC_CORE_ARGUMENT_R) to the command argument that is specified in bits 39 to 8 of the command format.
   d. Transfer mode register (SDHC_CORE_XFER_MODE_R). The host driver determines the multi / single block select, block count enable, data transfer direction, Auto CMD12 enable, and DMA enable. If the response check is enabled (RESP_ERR_CHK_ENABLE = '1'), set the response interrupt disable (RES_INT_DISABLE) to '1' and select the response type (RESP_TYPE) R1 (0: Memory) or R5 (1: SDIO).
2. Set the command to be executed into the command register (SDHC_CORE_CMD_R).
3. If response check is enabled, go to Step (6); else wait for the command complete interrupt.
4. Write '1' to clear the command complete bit (SDHC_CORE_NORMAL_INT_STAT_R.CMD_COMPLETE) of the normal interrupt status register.
5. Read the response register (SDHC_CORE_RESPXX_R) to get the necessary response to the command issued.
6. If writing to a card, go to Step (7). If reading from a card, go to Step (11).
7. Wait for the buffer write ready interrupt.
8. Write '1' to clear the buffer write ready bit (SDHC_CORE_NORMAL_INT_STAT_R.BUF_WR_READY) of the normal interrupt status register.
9. Write the block data (according to the number of bytes specified in Step (1)) to the buffer data port register (SDHC_CORE_BUF_DATA_R).
10. Repeat Step (7) to (9) until all blocks are sent and then go to Step (15).
11. Wait for the buffer read ready interrupt.
12. Write '1' to clear the buffer read ready bit (SDHC_CORE_NORMAL_INT_STAT_R.BUF_RD_READY) of the normal interrupt status register.
13. Read the block data (according to the number of bytes specified in Step (1)) from the buffer data port register (SDHC_CORE_BUF_DATA_R).
14. Repeat Step (11) to (13) until all blocks are received, and then go to Step (15).
15. If this sequence is for the single- or multi-block transfer, go to Step (16). In the case of the infinite block transfer, go to Step (18).
16. Wait for the transfer complete interrupt.
17. Write '1' to clear the transfer complete bit (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE) of the normal interrupt status register.
18. Perform the sequence for abort transaction in accordance with the Abort Transaction Sequence.

## 2.2 Data Transaction Sequence with Single Operation DMA (SDMA)

Figure 2 shows the data transaction sequence with SDMA.

Figure 2. Data Transaction Sequence with SDMA




The points highlighted in the above flowchart are described as follows:

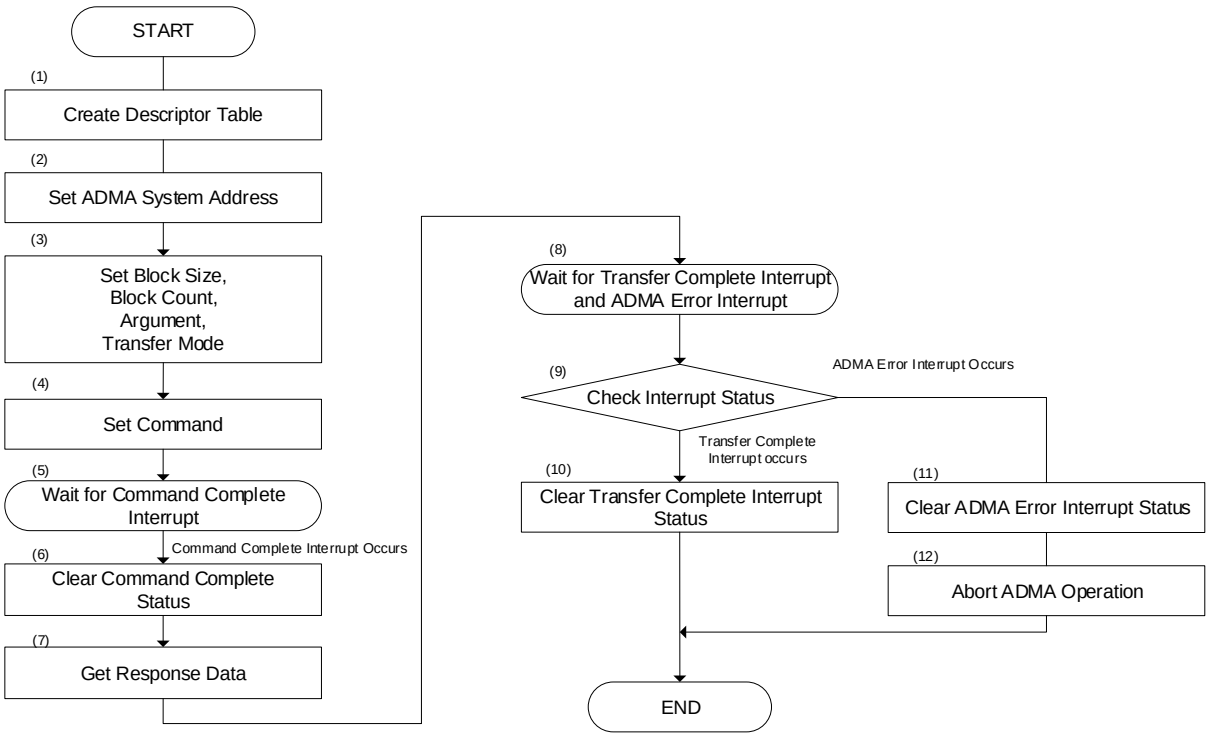
1. Set the data location of the system memory to the SDMA system address register (SDHC_CORE_SDMASA_R) if the host version 4 enable (SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE) = '0' or set to the ADMA system address register (SDHC_CORE_ADMA_SA_LOW_R) if the host version 4 enable (SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE) = '1'.
2. Configure the following entities:
   a. Block size register (SDHC_CORE_BLOCKSIZE_R) to the required length of a block.
   b. Block count register to the required data block count. If SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE = '0' or the 16-bit block count register (SDHC_CORE_BLOCKCOUNT_R) is set to non-zero, the 16-bit block count register is selected. If SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE = '1' and the 16-bit block count register is set to 0x0000, the 32-bit block count register (SDHC_CORE_SDMASA_R) is selected.
   c. Argument register (SDHC_CORE_ARGUMENT_R) to the command argument that is specified in bits 39 to 8 of the command format.
   d. Transfer mode register (SDHC_CORE_XFER_MODE_R). The host driver determines the multi / single block select, block count enable, data transfer direction, Auto CMD12 enable, and DMA enable. If response check is enabled (RESP_ERR_CHK_ENABLE = '1'), set the response interrupt disable (RES_INT_DISABLE) to '1' and select the response type (RESP_TYPE) R1 (0: Memory) or R5 (1: SDIO).
3. Set the command to be executed into the command register (SDHC_CORE_CMD_R).
4. If response check is enabled, go to Step (7); else wait for the command complete interrupt.

5. Write ‘1’ to clear the command complete bit (SDHC_CORE_NORMAL_INT_STAT_R.CMD_COMPLETE) of the normal interrupt status register.
6. Read the response register (SDHC_CORE_RESPXX_R) to get the necessary response to the command issued.
7. Wait for the data transfer complete interrupt (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE) and DMA interrupt (SDHC_CORE_NORMAL_INT_STAT_R.DMA_INTERRUPT).
8. If the data transfer complete bit (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE) is set to ‘1’, go to Step (11); else if the DMA interrupt bit (SDHC_CORE_NORMAL_INT_STAT_R.DMA_INTERRUPT) is set to ‘1’, go to Step (9). The data transfer complete interrupt has a higher priority than the DMA interrupt.
9. Write ‘1’ to clear the DMA interrupt bit (SDHC_CORE_NORMAL_INT_STAT_R.DMA_INTERUPT) in the normal interrupt status register.
10. Update the system address register (SDHC_CORE_SDMASA_R or SDHC_CORE_ADMA_SA_LOW_R) and go to Step (7).
11. Write ‘1’ to clear the data transfer complete interrupt bit (XFER_COMPLETE) and the DMA interrupt bit (DMA_INTERUPT) of the normal interrupt status register (SDHC_CORE_NORMAL_INT_STAT_R).

## 2.3 Data Transaction Sequence with Advanced DMA (ADMA)

Figure 3 shows the data transaction sequence with ADMA.

Figure 3. Data Transaction Sequence with ADMA




The points highlighted in the above flowchart are described as follows:

1. Create the descriptor table for ADMA in the system memory.
2. Set the descriptor address for ADMA in the ADMA system address register (SDHC_CORE_ADMA_SA_LOW_R).
3. Configure the following entities:
   a. Block size register (SDHC_CORE_BLOCKSIZE_R) to the required length of a block.
   b. Block count register to the required data block count. If SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE = '0' or the 16-bit block count register (SDHC_CORE_BLOCKCOUNT_R) is set to non-zero, the 16-bit block count register is selected. If SDHC_CORE_HOST_CTRL2_R.HOST_VER4_ENABLE = '1' and the 16-bit block count register is set to 0x0000, the 32-bit block count register (SDHC_CORE_SDMASA_R) is selected.
   c. Argument register (SDHC_CORE_ARGUMENT_R) to the command argument that is specified in bits 39 to 8 of the command format.
   d. Transfer mode register (SDHC_CORE_XFER_MODE_R). The host driver determines the multi / single block select, block count enable, data transfer direction, Auto CMD12 enable, and DMA enable. If the response check is enabled (RESP_ERR_CHK_ENABLE = '1'), set the response interrupt disable (RES_INT_DISABLE) to '1' and select the response type (RESP_TYPE) R1 (0: Memory) or R5 (1: SDIO).
4. Set the command to be executed into the command register (SDHC_CORE_CMD_R).
5. If the response check is enabled, go to Step (8); else wait for the command complete interrupt.
6. Write '1' to clear the command complete interrupt bit (SDHC_CORE_NORMAL_INT_STAT_R.CMD_COMPLETE) of the normal interrupt status register.

7. Read the response register (SDHC_CORE_RESPXX_R) to get the necessary response to the issued command.
8. Wait for the transfer complete interrupt and ADMA error interrupt.
9. If the transfer complete bit is set to ‘1’ (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE), go to Step (10); or else if the ADMA error interrupt bit is set to ‘1’ (SDHC_CORE_ERROR_INT_STAT_R.ADMA_ERR), go to Step (11).
10. Write ‘1’ to clear the transfer complete interrupt status bit (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE) of the normal interrupt status register.
11. Write ‘1’ to clear the ADMA error interrupt status bit (SDHC_CORE_ERROR_INT_STAT_R.ADMA_ERR) of the error interrupt status register.
12. Abort the ADMA operation. SD card operation should be stopped by issuing the abort command. The host driver can check the ADMA error status register (SDHC_CORE_ADMA_ERR_STAT_R) to analyze the generated ADMA error.

# 3 Abort Transaction Sequence

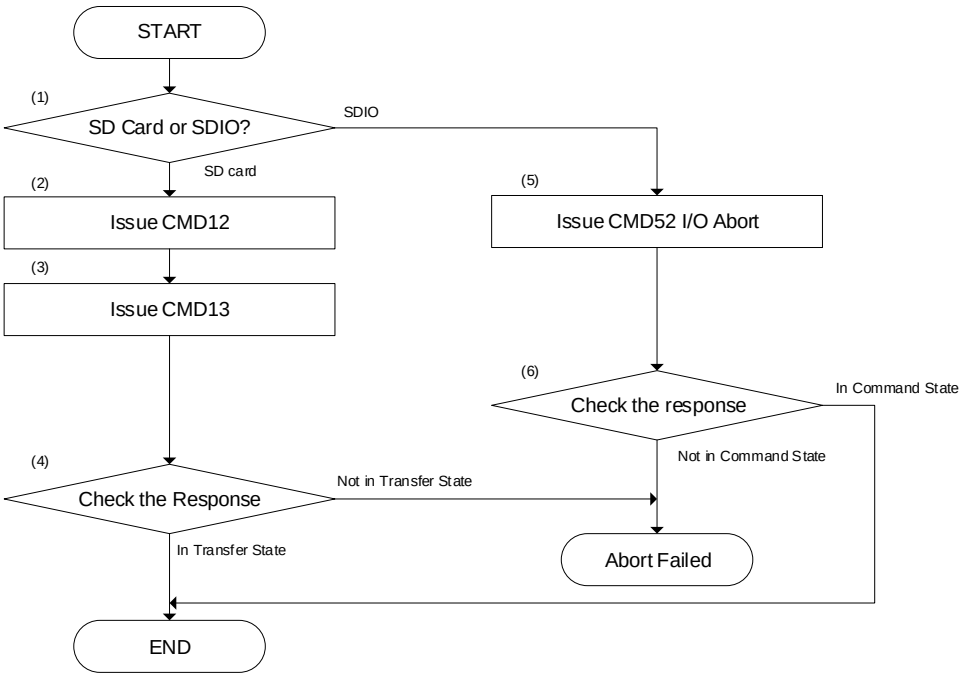
An abort transaction is performed by issuing CMD12 for an SD card and CMD52 for SDIO. There are two conditions under which the host driver needs to perform an abort transaction. The first condition is when the host driver wants to stop infinite block transfers and the second one is when the host driver wants to stop the transfers while multiple block transfers are in progress.

There are two methods to issue an abort command: asynchronous and synchronous. In an asynchronous abort sequence, the host driver can issue an abort command at any time unless ‘command inhibit’ in the present state register is set to ‘1’ (SDHC_CORE_PSTATE_REG.CMD_INHIBIT). In a synchronous sequence, the host driver will issue an abort command after the data transfer is stopped by setting the stop transaction at the block gap request bit (SDHC_CORE_BGAP_CTRL_R.STOP_BG_REQ) of the block gap control register to ‘1’.

## 3.1 Abort Command Sequence

Figure 4 shows the abort command sequence.

Figure 4. Abort Command Sequence



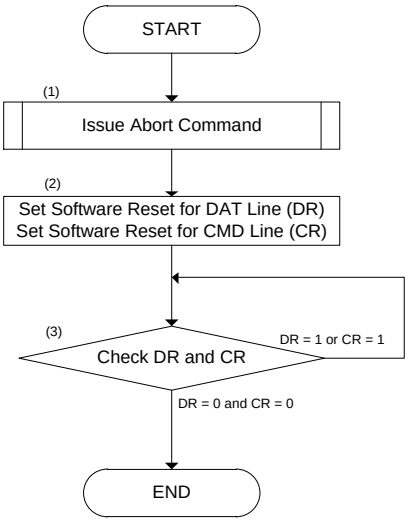
The points highlighted in the above flowchart are described as follows:

1. Check whether the connected device is SD card or SDIO. Steps (2) to (4) for card abort and Steps (5) to (6) for I/O abort.
2. Issue CMD12 for the card abort. If the card is already in transfer state, CMD12 is not accepted; the host driver needs to check the card state in Step (3).
3. Issue CMD13 to check the card state after completion of CMD12.
4. Check the response, and if the card is in transfer state, abort succeeds. Otherwise, abort fails.
5. Issue CMD52 I/O abort with RAW (read after write).
6. Check the response. If the SDIO card is in command state, abort succeeds. Otherwise, abort fails.

## 3.2 Asynchronous Abort Sequence

Figure 5 shows the asynchronous abort sequence.

Figure 5. Asynchronous Abort Sequence



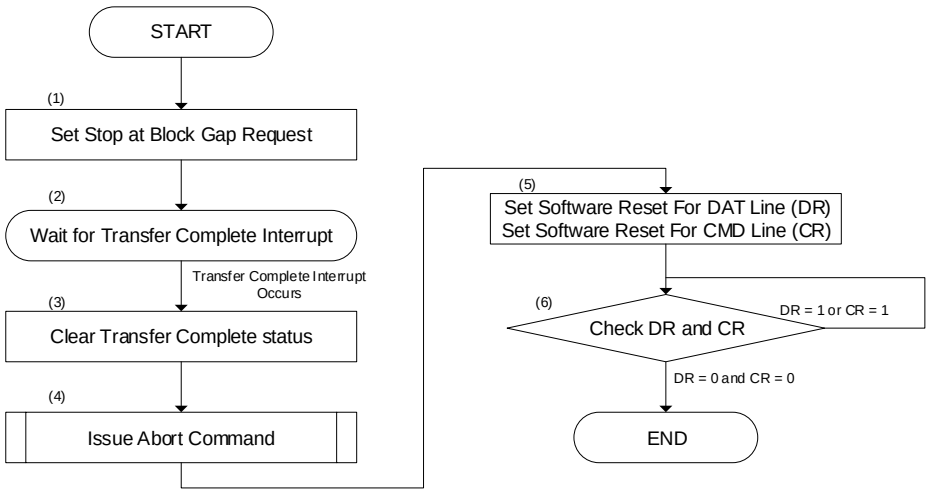

The points highlighted in Figure 5 are described as follows:

1. Issue an abort command.
2. To discard the data in the host controller buffer, set the software reset for the DAT line (SW_RST_DAT) to ‘1’ in the software reset register (SDHC_CORE_SW_RST_R). If the abort command of Step (1) is completed successfully, the command circuit that is reset by the software reset for the CMD line (SW_RST_CMD) in the software reset register (SDHC_CORE_SW_RST_R) is not needed.
3. Wait for completion of all software resets executed in Step (2) which happen when the bits which are set to ‘1’ in Step (2) are cleared to ‘0’.

## 3.3 Synchronous Abort Sequence

Figure 6 shows the synchronous abort sequence.

Figure 6. Synchronous Abort Sequence




The points highlighted in the above flowchart are described as follows:

1. Set the stop transaction at the block gap request bit (SDHC_CORE_BGAP_CTRL_R.STOP_BG_REQ) of the block gap control register to ‘1’ to stop SD transactions.
2. Wait for the transfer complete interrupt.
3. Write ‘1’ to clear the transfer complete bit (SDHC_CORE_NORMAL_INT_STAT_R.XFER_COMPLETE) of the normal interrupt status register.
4. Issue the abort command. If the SD clock has been stopped to halt the read operation, the host controller provides the SD clock to be able to issue the abort command, but the data circuits including DMA are still stopped.
5. To discard the data in the host controller buffer, set the software reset for DAT line (SW_RST_DAT) to ‘1’ in the software reset register (SDHC_CORE_SW_RST_R). If the abort command of Step (4) is completed successfully, the command circuit that is reset by the software reset for the CMD line (SW_RST_CMD) in the software reset register (SDHC_CORE_SW_RST_R) is not needed.
6. Wait for completion of all software resets executed in Step (5) until the bits which have been set to ‘1’ in step (5) are cleared to ‘0’.

## 3.4 Reset Command

The host driver should use a reset command (CMD0) when a communication between host and card is not recovered through the abort transaction. Before issuing a reset command, execute the software reset for the DAT line and the software reset for the CMD line to reset the command and data circuits of the host controller.

# 4 Glossary

| Terms | Description |
|---|---|
| ADMA | Advanced DMA |
| CMD12 | Command to abort multiple-block/infinite-block transfer operation |
| Auto CMD12 | Host Controller function to issue CMD12 automatically to abort multiple-block transfer operation |
| Block | Number of bytes, basic data transfer unit |
| Block Gap | Period between blocks of data |
| CMD | Command |
| DAT | Data bus |
| eMMC | embedded Multi-Media Card |
| SD | Secure Digital |
| SDHC | Secure Digital High Capacity |
| SDIO | Secure Digital Input Output |
| SDMA | Single Operation DMA |

# 5 Related Documents

The following are the Traveo™ II family series datasheets and technical reference manuals. Contact Technical Support to obtain these documents.

- Device datasheet
  - CYT3BB/4BB Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT4BF Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
- Body Controller High Family
  - Traveo™ II Automotive Body Controller High Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT3BB/4BB
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT4BF

# Document History

Document Title: AN224414 – Using SDHC Host Controller in Traveo II Family MCU

Document Number: 002-24414

| Revision | ECN | Submission Date | Description of Change |
|---|---|---|---|
| ** | 6681817 | 09/25/2019 | New application note. |
| *A | 6836593 | 03/24/2020 | Changed target parts number (CYT4 series).<br>Added target parts number (CYT3 series). |

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

## Products

| | |
|---|---|
| Arm® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Internet of Things | cypress.com/iot |
| Memory | cypress.com/memory |
| Microcontrollers | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| Power Management ICs | cypress.com/pmic |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless Connectivity | cypress.com/wireless |

## PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

## Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training | Components

## Technical Support

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709