AN184

Application Note

EP72XX INTERFACE TO MULTIMEDIACARD

P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.cirrus.com

AUG '00
AN184REV1

TABLE OF CONTENTS

LIST OF FIGURES

Contacting Cirrus Logic Support

For a complete listing of Direct Sales, Distributor, and Sales Representative contacts, visit the Cirrus Logic web site at:
http://www.cirrus.com/corporate/contacts/

1. INTRODUCTION

This application note describes a simple interface between a Cirrus Logic EP72XX microcontroller and a MultiMediaCard.

The MultiMediaCard (MMC) is a universal low-cost data storage and communication media. It is designed to cover a wide area of applications such as electronic toys, organizers, PDAs, cameras, smart phones, digital recorders, and digital players (such as MP3).

The MMC communication is based on a 7-pin serial bus designed to operate in a low voltage range. A MMC communication protocol is defined as part of the standard and is referred to as MMC mode. For compatibility, an alternate communication protocol (a subset of MMC) is based on the SPI™ (Serial Peripheral Interface) standard. This application note will describe a method to support both modes.

2. BACKGROUND

For digital audio applications such as portable MP3 players, the digital content is typically stored in NAND Flash memory. The MMC offers an alternative storage media and has the advantage of being removable and portable. For operation using the SPI protocol, the host must be able to read data at a typical MP3 data rate of 128 kilo-bits-per-second (kbits/s).

For programming the MMC, 128 kbits/s rate is insufficient to provide short download times. A typical portable MP3 system may include a USB port that accepts MP3 data files from a host PC. Since USB is a relatively fast interface with bit rates approaching 5 Mbits/s average rate (due to operating system overhead), the main limitation to download speed is the speed at which the MMC can be programmed. First versions of MMC cards can be programmed at ~1 Mbits/s, or around 125 kbytes/s. Newer generations of MMC cards are expected to reach 2.5 Mbits/s, or ~400 kbytes/s

The EP72xx microcontrollers have a speed-limited SPI port. Though adequate for reading content (at least up to 128 kbits/s), it is too slow for writing. However, the high speed of the ARM® processor allows one to create a “virtual peripheral” to support both SPI and MMC modes. A software “bit-bang” method provides the necessary signals for data and command, while a chip select provides the clock. Transfer speed using this method approaches 3 Mbits/s. For future applications, the SD MMC (secure MMC) bus has added 3 more data lines so that 4-bits can be transferred per clock (instead of one). This should eventually improve download speeds approaching that of traditional NAND flash.

3. MMC MODE VIRTUAL PERIPHERAL SOLUTION

Two GPIO pins are used for MMC_Data and MMC_Command. A spare chip select line is used for MMC_Clock. Section 3.1, “Sample Code: MMC Interface Connection” contains sample code that illustrates a possible connection to the MMC interface. In this case, GPIO pin PA7 is used for MMC_Data and PA6 is used for MMC_Command. A schematic is shown in Figure 1.

To maximize transfer speed, the routine to send bytes to the MMC card should be written in assembly language. For illustrative purposes, the sample code is written in C with in-line assembly. Note that the code is "unrolled", i.e. it does not use a loop counter to set the number of bits transmitted (in this case, one byte or 8 bits). Doing so adds substantially more overhead. As shown, it takes 23 instructions to execute the main routine. This routine, when running out of cache, should be able to burst at 74/23 MHz, or approximately 3 Mbs (see Figure 2). Using a loop counter will add two extra instructions per iteration, plus another cycle for the conditional branch, for a total of 8 * (3 + 2) = 40 cycles, or about ½ the speed of the unrolled loop.

NOTE: The nCS5 byte field in MEMCFG2 = 0x3E. Chip select 5 is not cached and the write buffer is disabled.)

3.1 Sample Code: MMC Interface Connection

```
// Routine to send a bit stream to MMC card.
// Data to send is in R0, address of nCS5 is in R1, and GPIO address is in R2.
// For this example, R1=0x5000.0000, R2=0x8000.0000 (port A data register).
// It is also assumed that PA7 is an output and pins PA[6:0] are input prior
// to calling this function. Data is sent MSB first.
void
send_mmc_byte(char mmc_data, int port, int GPIO)
{
   __asm
      {
      strb     r0, [r2]          // get value and place in GPIO
      strb     r0, [r1]          // strobe cs5
      mov      r0, r0, lsl #1    // shift next bit left which places it in PA7.
      strb     r0, [r2]          // Repeat 7 more times.
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      mov      r0, r0, lsl #1
      strb     r0, [r2]
      strb     r0, [r1]
      }
}
```

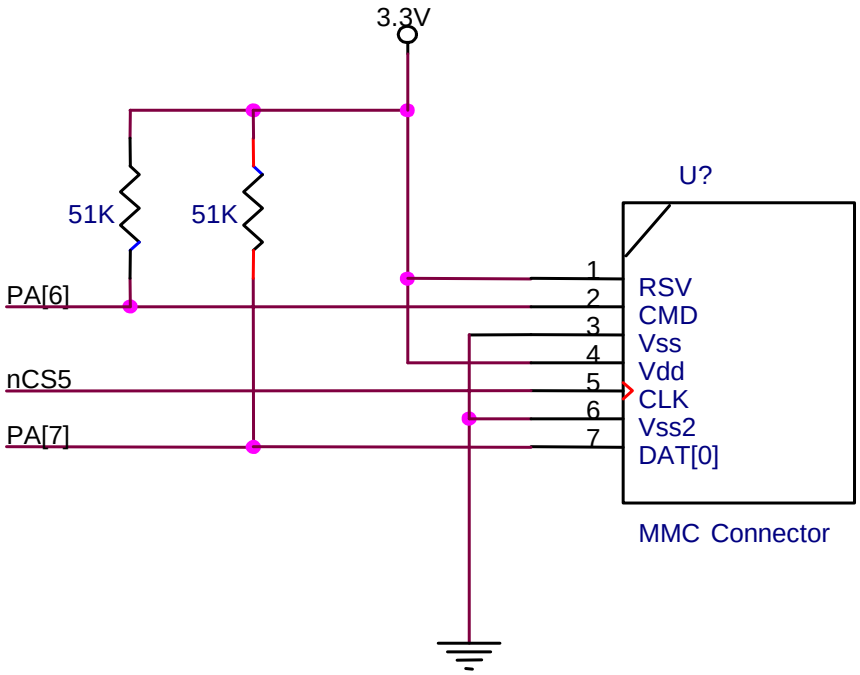

Figure 1. MMC in MMC mode

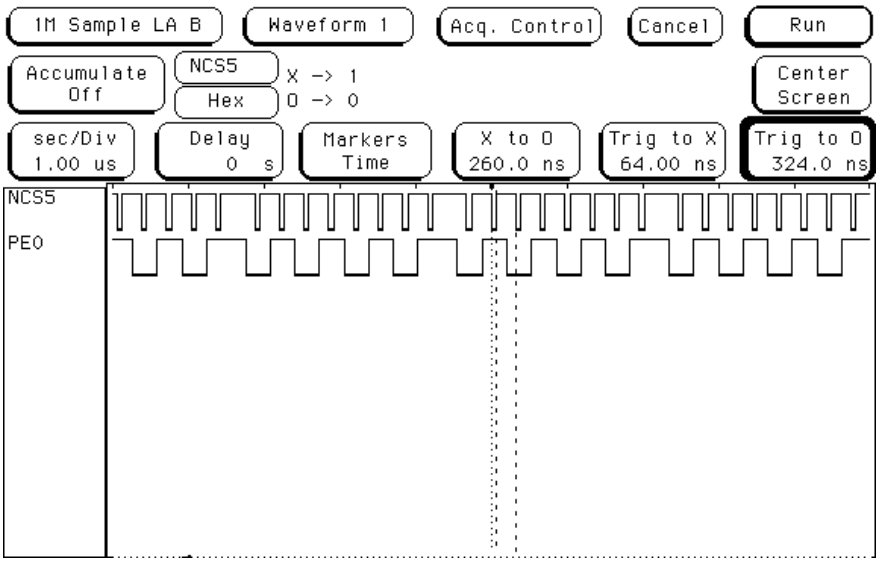

Figure 2. Waveform Measurement Results of send_mmc_byte.

4. SPI MODE VIRTUAL PERIPHERAL

To implement a SPI port, three GPIO signals are needed: chip select, data out, and data in. As with the MMC interface, nCS5 is used for the SPI data clock. A schematic is shown in Figure 3. A C subroutine with in-line assembly to write an 8-bit byte of data to the MMC is shown below. Unlike the previous sample code example, the routine in Section 4.1, “Sample Code: Sending Data to MMC Using SPI Mode” uses a loop counter. Better performance can be achieved if the loop is unrolled. This is left as an exercise for the reader.

Both Virtual Peripheral examples were written using the ARM Tools Version 2.50. Code results were benchmarked on a Cirrus Logic EP7209 evaluation board running the Angel Debug Monitor.

In some cases, it is desired to be able to support both modes of MMC operation, i.e., SPI and MMC modes. In this case, it is necessary to provide a power switch to the MMC power supply. Upon power up, the MMC defaults to MMC mode. To set for SPI mode, a command is issued to the card. Once in SPI mode, the MMC cannot revert back to MMC Mode until power is recycled.

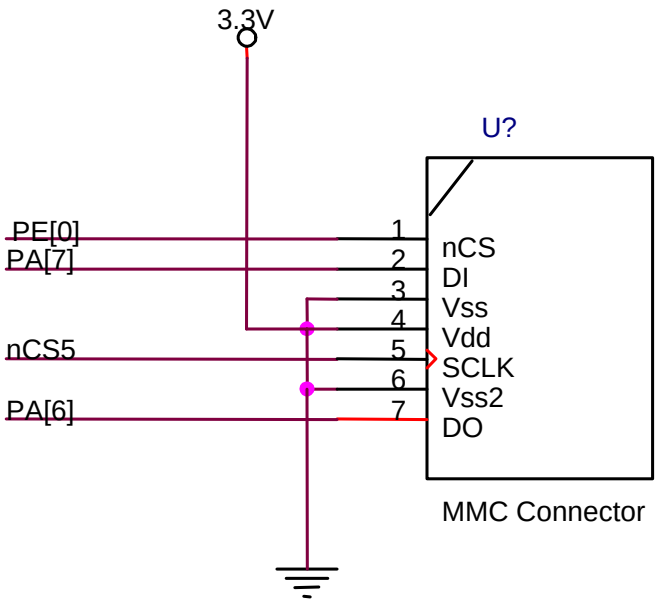

Figure 3. MMC in SPI Mode

4.1 Sample Code: Sending Data to MMC Using SPI Mode

```
// Routine to send a bit stream to the MMC card using the SPI mode.
// Data to send is in R0, address of nCS5 in R1 (in this case, 0x5000.0000),
// and R2 contains address of GPIO (Port A).
// It is also assumed that PA7 is an output and pins PA[6:0] are input prior
// to calling this function. In addition, the MMC chip select must be
// asserted (set to low) prior to using this routine. If a PA bit is used
// for CS, added instructions and registers are required to mask off the CS
// bit and will affect transfer speed.
// Data is sent MSB first.
   void
   send_spi_byte(char spi_data, int port, int GPIO)
   {
      __asm
      {
      mov  r3,     #0x8              //loop counter
      s1:
      strb r0,     [r2]              // output bit to PA7
      strb r0,     [r1]              // and strobe nCS5
      mov          r0, r0, lsl #1    // get next bit by shifting left once
      subs r3,     r3, #1            // are all bits transferred?
      bne          s1                // no.  branch back
      }
   }
```

5. CONCLUSION

Using the fast instruction set of the ARM7 and by careful design of software, it is possible to generate a software virtual peripheral that can interface to a MMC memory card using either the MMC or SPI interface protocol. Speeds up to 3 Mbit/s can be achieved in this manner, which is adequate for most applications. For best performance, follow these guidelines:

- Data read or written should use the multiple block mode for data transfer. Some MMC cards will not support multiple block mode with the SPI interface
- Use MMC mode if possible.
- Be aware that it can take as long as 2.5 mS after a command is sent to the MMC before it is ready to send data. This is referred to as the N_{AC} access time delay.
- Keep a flat file structure on the MMC so that blocks are contiguous thereby minimizing N_{AC}
- Unroll all assembly loops
- Try to design with the fastest and widest program memory possible. This will improve overall speed.

6. REFERENCES

Information on the MMC specifications can be found at the MultiMediaCard Association Web site www.mmca.org.

Data sheets for MMC cards can be found at several locations. Start by trying www.sandisk.com.

Information on Cirrus Logic's ARM-based microcontrollers, including data sheets on the EP7209/12 can be found at www.cirrus.com.

• Notes •


Maverick™