

EE445M/EE380L.12

Embedded and Real-Time Systems/ Real-Time Operating Systems

Lecture 7: Secure Digital Card, DMA

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

1

Secure Digital Card (SDC)

- Memory card standard
 - Upwards-compatible to multi-media card (MMC)
 - Reduced-size variants (miniSD, microSD)
 - Embedded micro-controller
 - Block based access (512 bytes/block)
 - Usually FAT file system

Source: http://elm-chan.org/docs/mmc/mmc_e.htmlOther references: <http://www.sdcard.org/home>http://www.ece.utexas.edu/~valvano/EE345M/SD_Physical_Layer_Spec.pdf

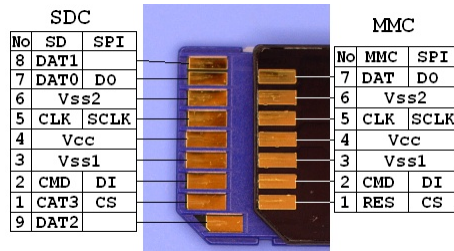
Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

2

SD Card Interfacing

- Native SD/MMC mode or SPI
 - 4-bit and 1-bit native modes
 - 9 SDC pads (3 for power supply, 6 effective)
 - 2.7 to 3.6V power supply
 - Up to 15mA standby
 - Up to 50-100mA in write mode



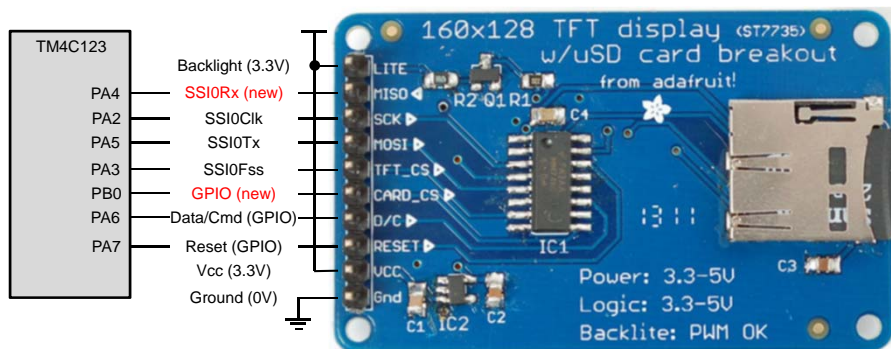
Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

3

ST7735 SDC Connector

- Using TM4C123 SPI interface
 - Two SSI0 slaves (TFT & Card via chip select)



Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

4

Serial Peripheral Interface (SPI)

- Serial on-board, inter-IC connection
 - Motorola (Freescale)
 - Similar to I²C (Philips)
 - 3-4 wires, up to 20Mbps

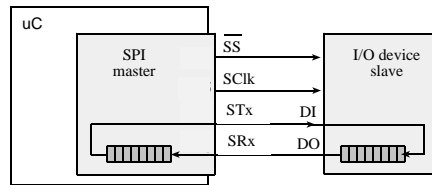
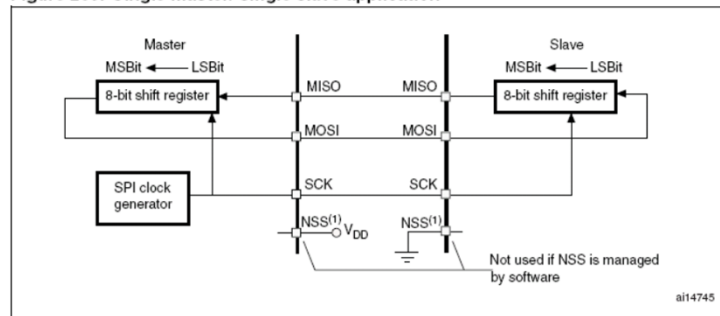


Figure 209. Single master/ single slave application



SPI Physical Layer Protocol

- Synchronous (shared clock) protocol
 - Shift and latch on opposite clock edges
 - Four operating modes
 - Clock polarity (CPOL/SPO) and phase (CPHA/SPH)
 - SDC uses Mode 0 (CPOL=0, CPHA=0)

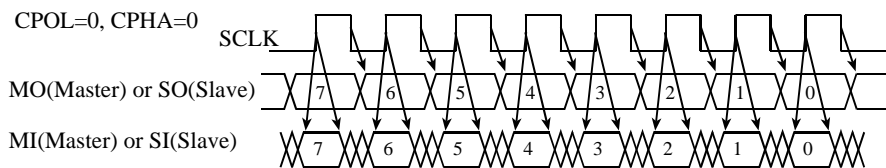


Figure 6.14. SPI CPOL= 0, CPHA=0 mode.

SPI Command & Response

- Serial, byte-oriented communication
 - Fixed length (6 bytes) command frame packet
 - Host to device: CMD(1 byte), Arg(4 byte), CRC(1 byte)
 - Up to 8 bytes command response time (NCR)
 - Host continues to read & send (0xFF) bytes
 - 1 or more bytes response (R1, R2, or R3/R7)

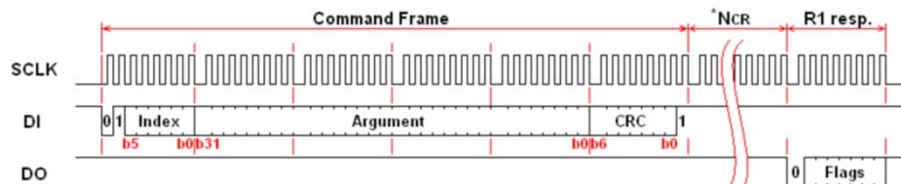


Figure 6.13. SD command frame.

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

7

SPI Command

Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read Operations Condition Register (OCR). Indicates supported working voltage range.

*1: ACMD<n> means a command sequence of CMD55-CMD<n>.

*2: Rsv(0)[31], HCS[30], Rsv(0)[29:0]

*3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]

Lecture 7

Table 6.6. SD commands.

8

SPI Response

Figure 7-9: R1 Response Format

R1b Response:
Variable length busy flag
(DO held low)

Figure 7-11: R3 Response Format

Figure 7-10: R2 Response Format

Lecture 7

SDC Initialization Procedure

- To put SDC into SPI mode
 1. Power ON (Insertion)
 - After Vcc > 2.2V, wait for > 1ms
 - Set clock between 100 and 400 kHz
 - Set DI and CS high, send 74 or more clock pulses
 2. Software reset (Set to SPI mode)
 - Send CMD0 with CS low (and proper CRC)
 - Card enters SPI and responds with R1 idle state (0x01)
 3. Initialization (CMD0, CMD1/ACMD41, CMD58)
 - Send ACMD41 (SDCv1) or CMD1 (MMC)
 - Repeat until R1 response changes to 0x00 (100s of ms)
 - Increase clock rate (25Mhz or more)

Lecture 7

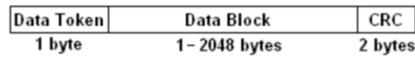
J. Valvano, A. Gerstlauer
EE445M/EE380L.12

10

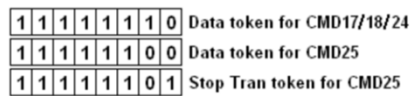
Data Transfer (1)

- Data packet and data response
 - Sent/received after command response
 - Data packet with token, data block, CRC
 - Token \$FE for read/single-write, \$FC/\$FD for multi-write

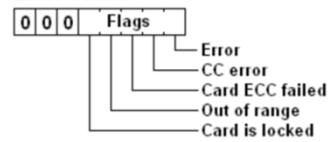
Data Packet



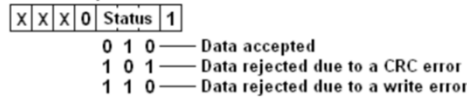
Data Token



Error Token



Data Response

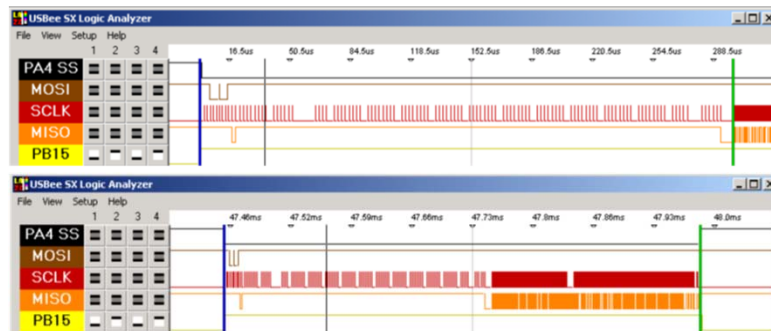
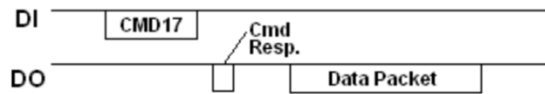


Lecture 7

11

Data Transfer (2)

- Single block read



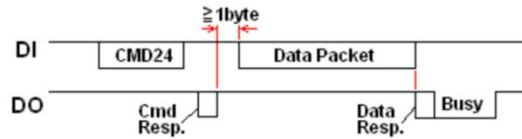
Lecture 7

It took 300µs to setup and 535µs to read one 512 byte block (logic analyzer resolution too slow for SCLK)

12

Data Transfer (3)

- Single block write



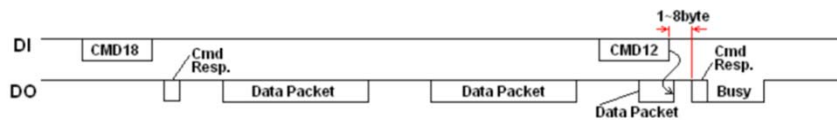
It took 272µs to write one 512 byte block
(logic analyzer resolution too slow for SCLK)

Lecture 7

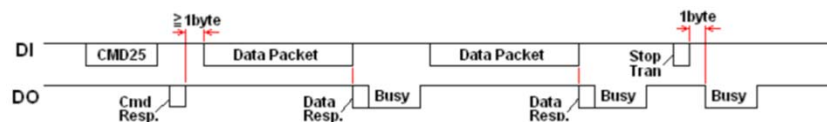
13

Data Transfer (4)

- Multi block read



- Multi block write



Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

14

SD Card Driver

```

// DSTATUS of type BYTE (8 bits)
// STA_NOINIT 0x01 Drive not initialized
// STA_NODISK 0x02 No medium in the drive
// STA_PROTECT 0x04 Write protected

DSTATUS eDisk_Initialize(BYTE drv);

DSTATUS eDisk_Status(BYTE drv);

// DRESULT of type BYTE (8 bits)
// RES_OK 0: Successful
// RES_ERROR 1: R/W Error
// RES_WRPRT 2: Write Protected
// RES_NOTRDY 3: Not Ready
// RES_PARERR 4: Invalid Parameter

DRESULT eDisk_Read (
  BYTE drv, // Physical drive number (0)
  BYTE *buff, // Pointer to buffer to read data
  DWORD sector, // Start sector number (LBA)
  BYTE count); // Sector count (1..255)

DRESULT eDisk_Write (
  BYTE drv, // Physical drive number (0)
  const BYTE *buff, // Pointer to the data to be written
  DWORD sector, // Start sector number (LBA)
  BYTE count); // Sector count (1..255)
)

//***** eDisk_ReadBlock *****
// Read 1 block of 512 bytes from the SD (write to RAM)
// Inputs: pointer to an empty RAM buffer
// sector number of SD card to read: 0,1,2,...
DRESULT eDisk_ReadBlock (
  BYTE *buff, /* Pointer to buffer to store data */
  DWORD sector /* Start sector number (LBA) */
)

//***** eDisk_WriteBlock *****
// Write 1 block of 512 bytes of data to the SD card
// Inputs: pointer to RAM buffer with information
// sector number of SD card to write: 0,1,2,...
DRESULT eDisk_WriteBlock (
  const BYTE *buff, /* Pointer to data to be written */
  DWORD sector /* Start sector number (LBA) */
)

```

**eDisc.h, eDisc.c
SDC_4C123.zip**

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

15

Benchmarking

Read performance (i=0..9999):

```

PD2 = 0x04;
eDisk_ReadBlock(buffer,block);
PD2 = 0x00;

```

Depends on

- Usage, mixed read-after-write
- SDC

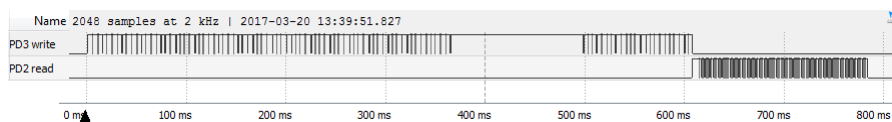
Write performance (i=0..9999):

```

PD3 = 0x08;
eDisk_WriteBlock(buffer,block);
PD3 = 0x00;

```

TM4C123 Kingston 2GB SD: SD-C02G
Write block time: 5500 μ s/block, 93 kB/s
Read block time: 1950 μ s/block, 262 kB/s



Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

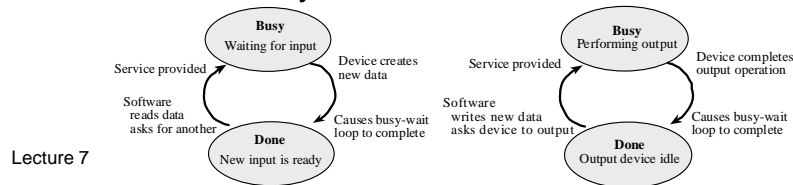
16

High-Speed Interfacing

- Bandwidth
 - Average or peak bytes transferred per second
- Latency
 - Interface latency

The time a need arises	The time the need is satisfied
new input is available	the input data is read
new input is available	the input data is processed
output device is idle	new output data is written
sample time occurs	ADC is triggered, input data
periodic time occurs	output data, DAC is triggered
control point occurs	control system executed

– Device latency

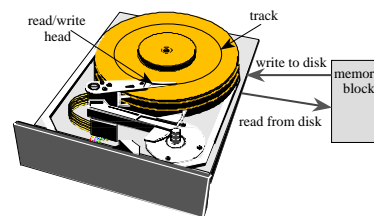


Lecture 7

17

High-Speed Applications (1)

- Mass storage
 1. Position head, wait for physical location (seek time)



2. Transfer data
 - 7200 RPM hard drive: 70 MB/s
 - SATA: up to 300 MB/sec
 - Original CD: 150 kB/s
 - 52x CD: 7.8 MB/s
 - 1x DVD: 1.4MB/s (9x CD)
 - 16x DVD: 22 MB/s (144x CD)
 - Class 2 SDC: 2 MB/s (13x CD)
 - Class 4 SDC: 4 MB/s (26x CD)



Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

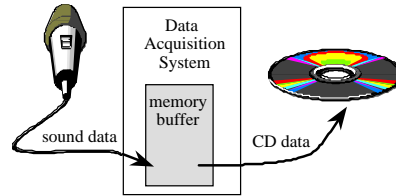
18

High-Speed Applications (2)

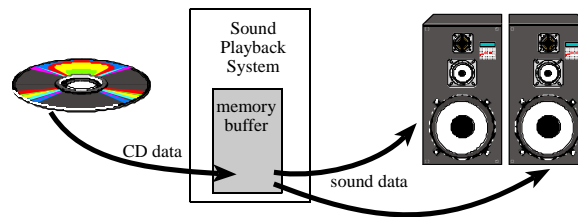
- High speed data acquisition

CD-quality sound
16-bit, 44kHz, stereo: 176 kB/s

Digital scope/signal generator
8-bit, 1GHz: 1 MB/s



- High-speed signal generation



Lecture 7

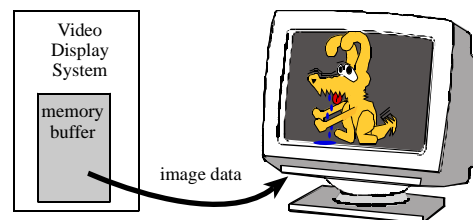
J. Valvano, A. Gerstlauer
EE445M/EE380L.12

19

High-Speed Applications (3)

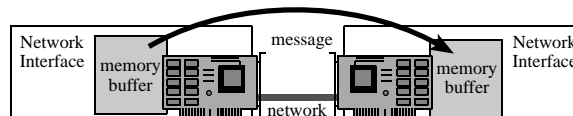
- Video displays

VGA display
256 colors (8-bit),
640x480, 60Hz: 18 MB/s



- Network communications

Ethernet
10Mbps: 1.2 MB/s



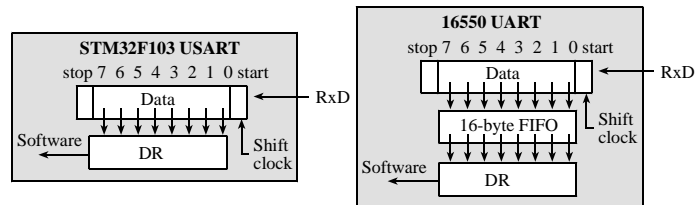
Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

20

High-Speed Interfaces (1)

- Hardware FIFO
 - Software satisfies average bandwidth, but not peak guarantees (max. latency)



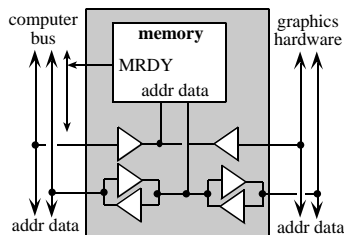
Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

21

High-Speed Interfaces (2)

- Dual-port memory
 - Shared memory between hardware & software
 - Framebuffer in video/graphics cards
 - Arbitrate between simultaneous accesses



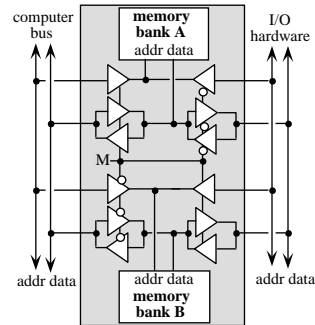
Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

22

High-Speed Interfaces (3)

- Bank-switched memory
 - Double-buffering
 - Share memory, but avoid conflicts
 - Alternate between different banks or buffers
 - Hardware accesses bank A/B
 - Software accesses bank B/A
 - Switch banks ($M=0/1$)

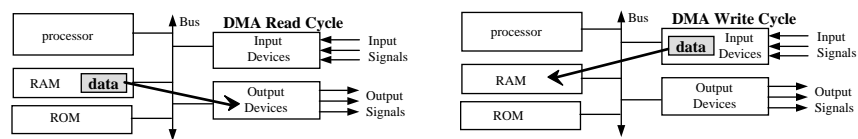


Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

23

Direct Memory Access (DMA)



- Transfer data directly
 - RAM/ROM \leftrightarrow device
- Does not involve software/processor
 - Frees up CPU to do other tasks
- At speed of device/memory

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

24

DMA Initiation

- DMA Controller (DMAC)
 - System bus master to handle DMA transactions
 - Software programmed (memory-mapped registers)
- Software initiated DMA
 - Software to setup DMA controller
 - Software triggers DMA transfer
 - Software to check for completion (poll/interrupt)
- Hardware initiated DMA
 - Software to setup DMA controller
 - Hardware triggers DMA transfer
 - Software to check for completion (poll/interrupt)

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

25

Burst vs. Cycle Steal DMA



Figure 6.6. An input block is transferred all at once during burst mode DMA.

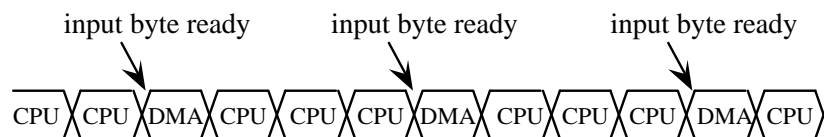


Figure 6.7. Each time an input byte is ready it is transferred to memory using single cycle DMA.

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

26

Dual Address DMA

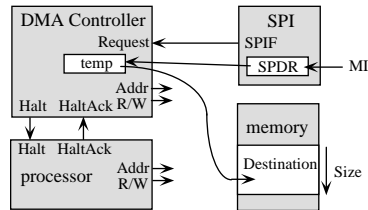


Figure 6.10. Block diagram showing the modules involved in a SPI read.

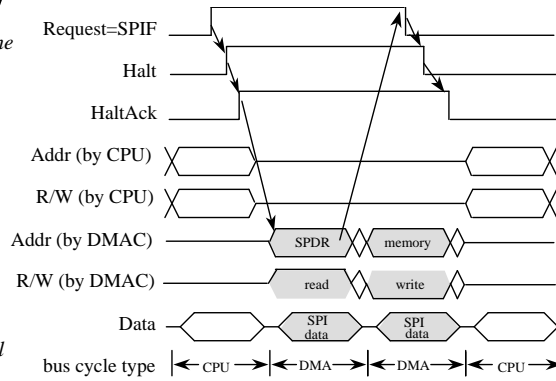


Figure 6.11. Timing diagram of a dual address DMA-controlled SPI read.

Single Address DMA

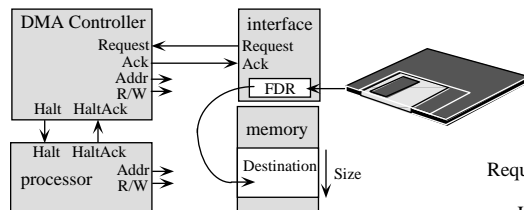


Figure 6.8. Block diagram showing the modules involved in a disk read.

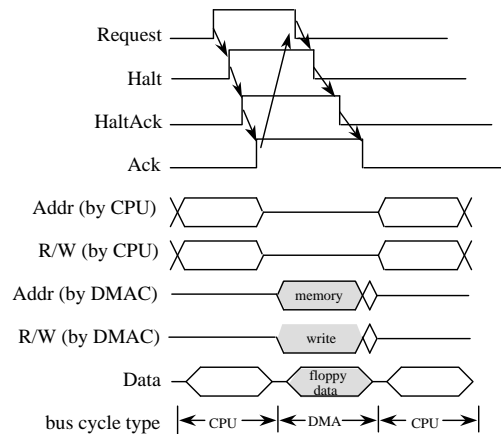
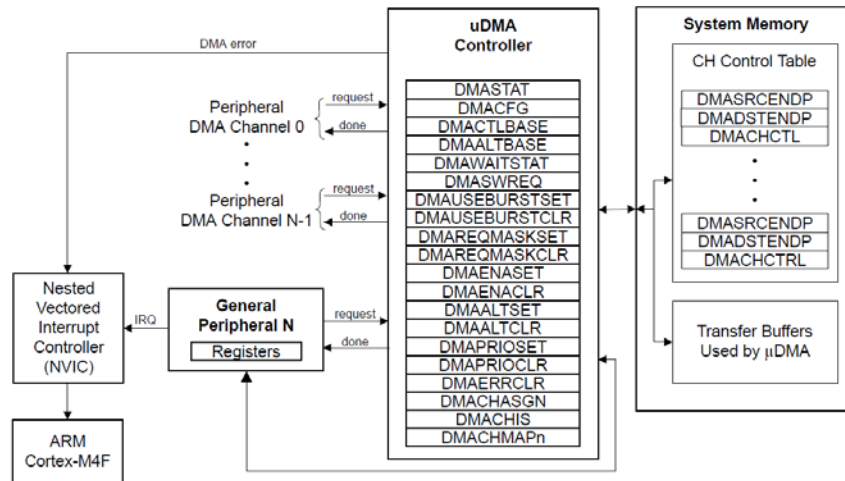


Figure 6.9. Timing diagram of a single address DMA-controlled floppy disk read.

TM4C123 DMA Programming

Figure 9-1. μ DMA Block Diagram



Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

29

DMA Channels

Table 9-1. μ DMA Channel Assignments (DMACHMAPn, High/Low Priority via DMAPRIOSET/DMAPRIOCLR)

Enc.	0		1		2		3		4	
	Peripheral	Type	Peripheral	Type	Peripheral	Type	Peripheral	Type	Peripheral	Type
0	USB0 EP1 RX	SB	UART2 RX	SB	Software	B	GPTimer 4A	B	Software	B
1	USB0 EP1 TX	B	UART2 TX	SB	Software	B	GPTimer 4B	B	Software	B
2	USB0 EP2 RX	B	GPTimer 3A	B	Software	B	Software	B	Software	B
3	USB0 EP2 TX	B	GPTimer 3B	B	Software	B	Software	B	Software	B
4	USB0 EP3 RX	B	GPTimer 2A	B	Software	B	GPIO A	B	Software	B
5	USB0 EP3 TX	B	GPTimer 2B	B	Software	B	GPIO B	B	Software	B
6	Software	B	GPTimer 2A	B	UART5 RX	SB	GPIO C	B	Software	B
7	Software	B	GPTimer 2B	B	UART5 TX	SB	GPIO D	B	Software	B
8	UART0 RX	SB	UART1 RX	SB	Software	B	GPTimer 5A	B	Software	B
9	UART0 TX	SB	UART1 TX	SB	Software	B	GPTimer 5B	B	Software	B
10	SSI0 RX	SB	SSI1 RX	SB	UART6 RX	SB	GPWideTimer 0A	B	Software	B
11	SSI0 TX	SB	SSI1 TX	SB	UART6 TX	SB	GPWideTimer 0B	B	Software	B
12	Software	B	UART2 RX	SB	SSI2 RX	SB	GPWideTimer 1A	B	Software	B
13	Software	B	UART2 TX	SB	SSI2 TX	SB	GPWideTimer 1B	B	Software	B
14	ADC0 SS0	B	GPTimer 2A	B	SSI3 RX	SB	GPIO E	B	Software	B
15	ADC0 SS1	B	GPTimer 2B	B	SSI3 TX	SB	GPIO F	B	Software	B
16	ADC0 SS2	B	Software	B	UART3 RX	SB	GPWideTimer 2A	B	Software	B
17	ADC0 SS3	B	Software	B	UART3 TX	SB	GPWideTimer 2B	B	Software	B
18	GPTimer 0A	B	GPTimer 1A	B	UART4 RX	SB	GPIO B	B	Software	B
19	GPTimer 0B	B	GPTimer 1B	B	UART4 TX	SB	Software	B	Software	B
20	GPTimer 1A	B	Software	B	UART7 RX	SB	Software	B	Software	B
21	GPTimer 1B	B	Software	B	UART7 TX	SB	Software	B	Software	B
22	UART1 RX	SB	Software	B	Software	B	Software	B	Software	B
23	UART1 TX	SB	Software	B	Software	B	Software	B	Software	B
24	SSI1 RX	SB	ADC1 SS0	B	Software	B	GPWideTimer 3A	B	Software	B
25	SSI1 TX	SB	ADC1 SS1	B	Software	B	GPWideTimer 3B	B	Software	B
26	Software	B	ADC1 SS2	B	Software	B	GPWideTimer 4A	B	Software	B
27	Software	B	ADC1 SS3	B	Software	B	GPWideTimer 4B	B	Software	B
28	Software	B	Software	B	Software	B	GPWideTimer 5A	B	Software	B
29	Software	B	Software	B	Software	B	GPWideTimer 5B	B	Software	B
30	Software	B	Software	B	Software	B	Software	B	Software	B
31	Reserved	B	Reserved	B	Reserved	B	Reserved	B	Reserved	B

DMA Channel Control Structure

- Two per channel in main memory
 - Primary array (DMACTLBASE)
 - Alternate (DMAALTBASE) for continuous ping-pong

Address of the last byte of the source buffer								
Address of the last byte of the destination buffer								
DSTINC	DSTSIZE	SRCINC	SRCSIZE	ARBSIZE	XFERSIZE	NXTUSE	XFERMODE	

Table 6.4. Structure of an entry in the channel control structure.

Parameter	Definition
Source address	Address of the module (memory or input) that generates the data
Destination address	Address of the module (memory or output) that accepts the data
DSTINC	Automatically +1/+2/+4/0 the destination address after each transfer
DSTSIZE	Destination data size (byte/halfword/word)
SRCINC	Automatically +1/+2/+4/0 the source address after each transfer
SRCSIZE	Source data size (byte/halfword/word)
ARBSIZE	Size of bursts between bus arbitrations (powers of 2)
XFERSIZE	Number of items to transfer
NXTUSE	Next use burst for scatter-gather transfers
XFERMODE	Transfer mode (auto-request, ping-pong, etc.) and transfer status

Lecture 7

J. Valvano, A. Gerstlauer
EE445M/EE380L.12

31

Software-Triggered Memory-to-Memory DMA

```
// The ucControlTable table must be aligned to a 1024 byte boundary.
uint32_t ucControlTable[256] __attribute__((aligned(1024)));
#define CH30 (30*4) // channel 30 for SW-triggered
#define BIT30 0x40000000
// *****DMA_Init*****
// Initialize the memory to memory transfer
// This needs to be called once before requesting a transfer
void DMA_Init(void){ volatile uint32_t delay;
  SYSCTL_RCGCDMA_R = 0x01; // uDMA Module Run Mode Clock Gating Control
  delay = SYSCTL_RCGCDMA_R; // allow time to finish
  UDMA_CFG_R = 0x01; // MASTEN Controller Master Enable
  UDMA_CTLBASE_R = (uint32_t)ucControlTable;
  UDMA_PRIOTCLR_R = BIT30; // default, not high priority
  UDMA_ALTCCLR_R = BIT30; // use primary control
  UDMA_USEBURSTCLR_R = BIT30; // responds to both burst and single
  UDMA_REQMASKCLR_R = BIT30; // allow controller to recognize requests
}
// *****DMA_Xfr *****
// Called to transfer words from source to destination
// Inputs: src is a pointer to the first element of the original data
// dest is a pointer to a place to put the copy
// cnt is the number of words to transfer (max is 1024 words)
void DMA_Xfr(uint32_t *src, uint32_t *dest, uint32_t cnt){
  ucControlTable[CH30] = (uint32_t)src+cnt*4-1; // last address
  ucControlTable[CH30+1] = (uint32_t)dest+cnt*4-1; // last address
  ucControlTable[CH30+2] = 0xAA00C002+(cnt-1)<<4; // Control Word
/* DMACTRL
  Bits Value Description
  DSTINC 31:30 2 32-bit destination address increment
  DSTSIZE 29:28 2 32-bit destination data size
  SRCINC 27:26 2 32-bit source address increment
  SRCSIZE 25:24 2 32-bit source data size
  reserved 23:18 0 Reserved
  ARBSIZE 17:14 3 Arbitrates after 8 transfers
  XFERSIZE 13:4 cnt-1 Transfer cnt items
  NXTUSEBURST 3 0 N/A for this transfer type
  XFERMODE 2:0 2 Use Auto-request transfer mode
*/
  UDMA_ENASET_R = BIT30; // uDMA Channel 30 is enabled.
  UDMA_SWREQ_R = BIT30; // software start, do not wait for completion
}
```

DMASoftware_4C123.zip

**DMASPI_4C123.zip
(continuous looping transfer triggered by timer)**

32