

No Midterm and Quiz 2 make ups, unless you have medical reasons (need doctor certificate).

Due to the fact that there were 3 students that had talked to me prior to the quiz about not being able to attend and I was not clear in my answers, there will be a [make-up Quiz 1](#).

Quiz 1 can be rewritten in the same day with Quiz 2.

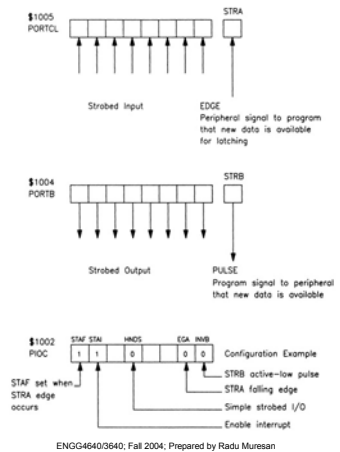
The time allocated for Quiz 1 and 2 will be 30min
 Quiz 1 will cover Chapter 1 & 2 of Spasov book plus the lecture notes for weeks 1, 2 and 3
 There will be 10 questions that will be different than the first quiz version.

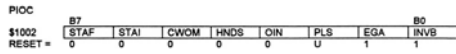
Parallel Input/Output

- A. Introduction to the subsystem
- B. Motorola's parallel input/output
- C. Seven-segment display output
- D. Liquid-crystal display
- E. Keyboard interfacing
- F. Other user input/output
- G. **Strobed input/output**
- H. Full handshake input/output
- I. Parallel interface standards

Strobed Input/Output

- Strobed input/output
 - input strobe
 - output strobe





STAF = Strobe A (STRA) Flag
 0 = inactive
 1 = Set at active edge of STRA pin
STAI = Strobe A Interrupt Enable
 0 = No hardware interrupt generated
 1 = Hardware interrupt requested when STAF = 1
CWOM = Port C Wire-OR Mode
 0 = Port C outputs normal
 1 = Open-drain
HNDS = Handshake/Simple Strobe Mode Select
 0 = Simple Strobe Mode
 1 = Full Handshake Mode
OIN = Output/Input Handshake Select
 0 = Input
 1 = Output
PLS = Pulse Mode Select for STRB Output
 0 = STRB level active
 1 = STRB pulses
EGA = Active Edge Select for STRA
 0 = High to Low (falling)
 1 = Low to High (rising)
INVB = Invert STRB Output
 0 = STRB active low
 1 = STRB active high

Input Strobe Program Example

```

ORG $100

LDX #REGBAS
CLR DDRC,X ;configure port C as input
LDAA #02 ;configure PIOC register
STAA PIOC,X ;active STRA is rising edge
LDY #PTR ;initialize storage pointer
* poll STAF for rising edge
CIN
BRCLR PIOC,X,$80,CIN
LDAA PORTCL,X
* ;latch in port C input
* ;when STRA edge detected
STAA 0,Y ;and store it
INY ;repeat 10 times
CPY #PTR+10
BNE CIN
STOP ;then stop
  
```

```
PTR EQU $180
```

* For assembly
 REGBAS EQU \$1000 ;Starting address for
 * ;register block

```

PAGE0 EQU 0
ORG 0

PORTA RMB 1 ;$00
RMB 1 ; Reserved
PIOC RMB 1 ;$02
PORTC RMB 1 ;$03
PORTB RMB 1 ;$04
PORTCL RMB 1 ;$05
RMB 1 ; Reserved
DDRC RMB 1 ;$07
PORTD RMB 1 ;$08
DDRD RMB 1 ;$09
  
```

Output Strobe Program Example

- Peripheral uses STRA to tell MCU that it needs data
- The MCU responds by writing a new byte to port B
 - this also pulses STRB for 2 E-clock periods to tell the peripheral that it has sent the requested data
- The example shows only the relevant part of the program code (see next slide)

* PIOC configured as shown in Figure 9.13
 * and all other required initialization has been done.

```

ORG    $100

LDY    #PTR    ;initialize data pointer
CLI    ;enable interrupts
REPEAT
WAI    ;wait for falling STRA
      ;to cause interrupt
*      LDAA    0,Y    ;send out data when it occurs
      STAA    PORTB,X
*      ;note, MCU also pulses STRB low for 2 E-cycles
      INY    ;repeat for next data transfer
      BRA    REPEAT
RIRQ
* This is the interrupt handler routine for parallel I/O
* Note it has the same vector ($FFF2,F3) as IRQ

LDAA    PIOC,X ;these two instructions clear STAF
LDAA    PORTCL,X
RTI
PTR    EQU    $180
  
```

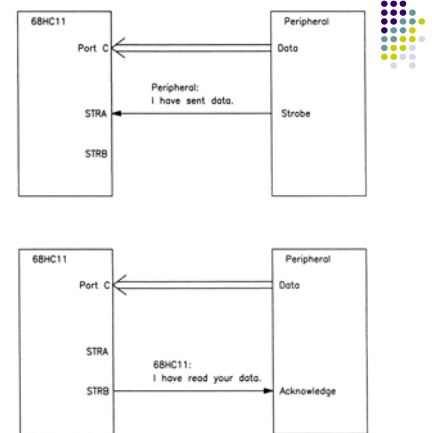
Parallel Input/Output

- A. Introduction to the subsystem
- B. Motorola's parallel input/output
- C. Seven-segment display output
- D. Liquid-crystal display
- E. Keyboard interfacing
- F. Other user input/output
- G. Strobed input/output
- H. Full handshake input/output
- I. Parallel interface standards

Full Handshake Input/Output

- The handshake protocol is an agreement whereby the receiver acknowledges each unit of data it receives
 - the transmitter waits for this acknowledgment before it sends the next unit
 - the unit can be a block or a byte
 - a parallel I/O subsystem handshake protocol transf. each byte as a unit
- 68HC11 supports automatic handshaking for parallel I/O for port C
 - this is determined by programming PIOC, specifically HNDC = 1
- 68HC11 can be set for:
 - input handshaking
 - pulsed operation
 - interlocked operation
 - output handshaking
 - pulsed operation
 - interlocked operations

Input handshake




```

* Subroutine INHNS
* Uses input handshake to read port C
* Before calling this subroutine, must call
* subroutine INITHNS to configure for handshake
* Calling Registers
*   IX = address of register block
* Return Registers
*   ACCA = data read from port C
*   Others unaffected except for CCR

INHNS
*   ;poll for STRA transition
BRCLR PIOC,X,$80,INHNS
LDAA PORTCL,X ;input strobed data and clear STAF
RTS ;return(data)

```

```

* For assembly
REGBAS EQU $1000 ;Starting address for
* ;register block

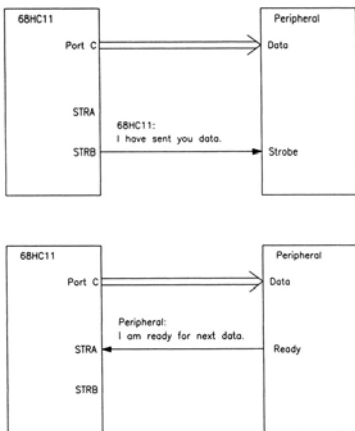
PAGE0 EQU 0
ORG 0

PORTA RMB 1 ;$00
RMB 1 ; Reserved

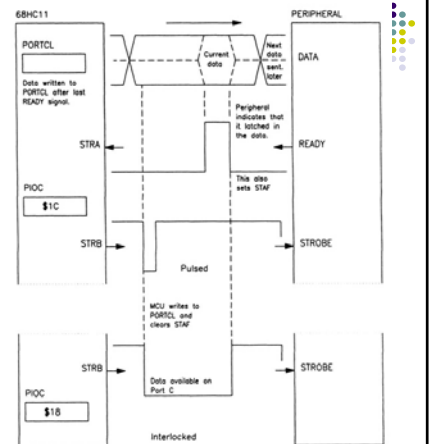
PIOC RMB 1 ;$02
PORTC RMB 1 ;$03
PORTB RMB 1 ;$04
PORTCL RMB 1 ;$05
RMB 1 ; Reserved
DDRC RMB 1 ;$07
PORTD RMB 1 ;$08
DDRD RMB 1 ;$09

```

Output Handshake



Pulsed Operation



Interlocked Operation

Output Handshake Example

- Program outputs same data when requested by peripheral
- subsystem configuration identical to the interlocked configuration of slide 15

```

        ORG   $100
DIR     EQU   $00
CONF    EQU   $01

OUTDATA EQU   $02 ;data stored here

DEMOOUT
LDX     #REGBAS ;point to registers
LDAA   #$FF ;configure for output direction
STAA   DIR
LDAA   #$18 ;and output handshake, interlocked
STAA   CONF
JSR    INITHNDS
REPEATOUT
LDAA   OUTDATA
JSR    OUTHNDS
BRA   REPEATOUT
    
```

* Subroutine INITHNDS, see Listing 9.13

- * Subroutine OUTHNDS
- * Uses output handshake to send data to port C
- * and waits for peripheral ready before returning
- * Calling Registers
- * IX = address of register block
- * ACCA = data to send out
- * No Return Registers except CCR affected

```

OUTHNDS
    STAA PORTCL,X ;output data, clears STAF also
* poll for STRA transition
OUTHNDS1
    ;by checking if STAF high
    BRCLR PIOC,X,$80 OUTHNDS1
    RTS ;and return, can now
* ;send next byte if desired
    
```

* For assembly

```

INITHNDS
* dummy address for subr

REGBAS EQU   $1000 ;Starting address for
                ;register block
PAGE0 EQU   0
        ORG   0

PORTA RMB   1 ;$00
        RMB   1 ;Reserved

PIOC  RMB   1 ;$02
PORTC RMB   1 ;$03
PORTB RMB   1 ;$04
PORTCL RMB   1 ;$05
        RMB   1 ;Reserved
DDRC  RMB   1 ;$07
PORTD RMB   1 ;$08
DDRD  RMB   1 ;$09
    
```

Parallel Input/Output

- A. Introduction to the subsystem
- B. Motorola's parallel input/output
- C. Seven-segment display output
- D. Liquid-crystal display
- E. Keyboard interfacing
- F. Other user input/output
- G. Strobed input/output
- H. Full handshake input/output
- I. **Parallel interface standards**

ENGG4640/3640; Fall 2004; Prepared by Radu Muresan

25

Parallel Interface Standards

- Centronics parallel interface
- The IEEE-488 general-purpose instrumentation bus
- Small computer system interface (SCSI)

ENGG4640/3640; Fall 2004; Prepared by Radu Muresan

26

Centronics Parallel Interface

- the centronics parallel interface is commonly used for printers
- Handshaking is required because a computer sends a sequence of bytes to the printer

ENGG4640/3640; Fall 2004; Prepared by Radu Muresan

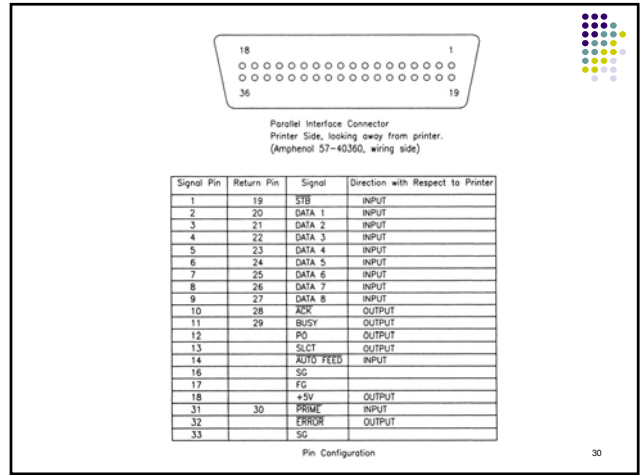
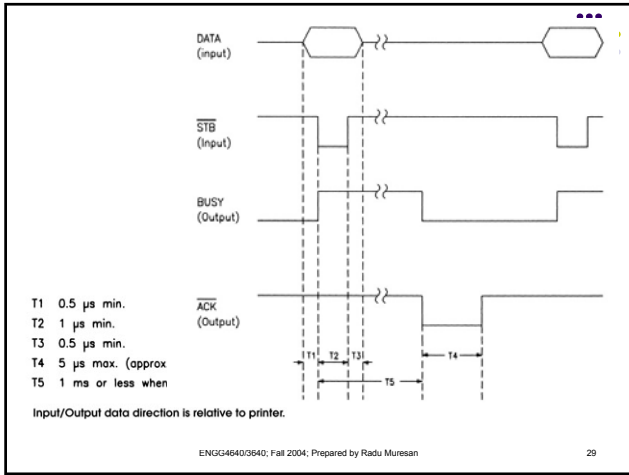
27

Centronics Parallel Interface Signal Description

- STB[^]: strobe pulse to read data
- DATA: D1 to D8 data
- ACK[^]: acknowledge. An approx. 5 μ s wide pulse to indicate that data has been received and printer is ready to accept new data
- BUSY: A high signal indicates that the printer can not receive data. It becomes H when:
 - data entry; printing operation; off-line state; error
- PO: paper out.
- SLCT: select state. H when on-line and L when off-line
- AUTOFEED[^]: when L a line feed is added automatically to a carriage return
- PRIME[^]: used to initialize the printer
- ERROR[^]: L when printer detect a fault
- SG: signal ground
- FG: frame ground
- +5V: TTL power supply

ENGG4640/3640; Fall 2004; Prepared by Radu Muresan

28



68HC as a Computer Sending Data to the Printer

- * I/O service routines to send output to "Centronics"
- * Printer Interface. Also demonstrates peripheral
- * interlocked output handshake. Calling program uses
- * subroutines to print data stored in a print spool buffer.

ENGG4640/3640; Fall 2004; Prepared by Radu Muresan 31

- * INTERFACE CONNECTIONS
 - * PRINTER MCU as computer
 - * DATA <----- Port C
 - * STB <----- STRB
 - * ACK -----> STRA
 - * BUSY -----> PD1
 - * PRIME <----- PD0
- * PIOC CONFIGURATION DETAILS
 - * INVb=0 STRB/STR active low
 - * EGA=0 STRA/ACK active low
 - * PLS=0 Interlock
 - * OIN=1 Output handshake
 - * HNDS=1 Full handshake mode
 - * CWOM=0 Normal CMOS outputs
 - * (LS compatible for Centronics standard)
 - * STAI=0 Disable interrupt
 - * STAF=x Sets on falling STRA line
 - * To clear STAF, read PIOC, then write PORTCL (when configured)
- * ASCII control characters
 - FMFD EQU \$0C ;ASCII FF character - formfeed, Ctrl-L
 - EOF EQU \$1A ;ASCII EOF character - end of file, Ctrl-Z

32

```
ORG $100
```

- * Subroutine INITPRN
- * Initializes printer interface
- * Calling Registers
- * IX = address of register block
- * No Return Registers except CCR affected

```
INITPRN
```

```
PSHA ;preserve registers
BSET DDRD,X $01 ;PD0 output, PRIME (reset) input
BCLR PORTD,X $01 ;PRIME low to initiate (reset) printer
BSET PORTD,X $01 ;PRIME high to enable printer
LDAA PIOC,X ;clear STAF if set
LDAA PORTCL,X
BSET DDRC,X $FF ;configure port C output
LDAA #$18 ;configure PIOC with
* ;magic number
STAA PIOC,X
PULA ;restore registers
RTS ;return
```

- * Subroutine PRINT_BYTE
- * Go here after ACK sets STAF during
- * calling program and makes STRB/STB active
- * Transmits addressed data byte to printer
- * unless end of file encountered in which case
- * it sends a formfeed to printer.

```
* Calling Registers
```

- * IX = address of register block
- * IY = address of data byte

```
* Return Registers
```

- * ACCA = printed character or \$0C (formfeed)
- * if end of file encountered
- * CCR affected

```
PRINT_BYTE
```

```
BUSY
```

```
BRSET PORTD,X $02 BUSY
* ;wait here if printer busy
LDAA 0,Y ;get character to print
CMPA #EOF ;if EOF character then stop printing
BEQ SFMFD
STAA PORTCL,X ;else send current char to printer
* ;also clears STAF
```

```
NOACK
```

```
BRCLR PIOC,X $80 NOACK
* ;if no ACK then wait here
* ;(until ACK sets STAF)
RTS ;otherwise return
```

```
* send formfeed to printer
```

```
SFMFD
```

```
* ; and return
```

```
LDAA #FMFD
STAA PORTCL,X
RTS
```

```
* For assembly
```

```
REGBAS EQU $1000 ;Starting address for
* ;register block
```

```
PAGE0 EQU 0
ORG 0
```

```
PORTA RMB 1 ;$00
RMB 1 ;Reserved
```

68HC11 as Printer

- Left as exercise

The IEEE-488 General-Purpose Instrumentation Bus



- Used for communication between a computer and instruments
 - Example: a computer tells a digital storage scope to sample data and to send the data to the computer for analysis
- The IEEE-488 interface is also an example of an interface in which open-collector (TTL) or open-drain (HCMOS) outputs are used

The IEEE-488 General-Purpose Instrumentation Bus

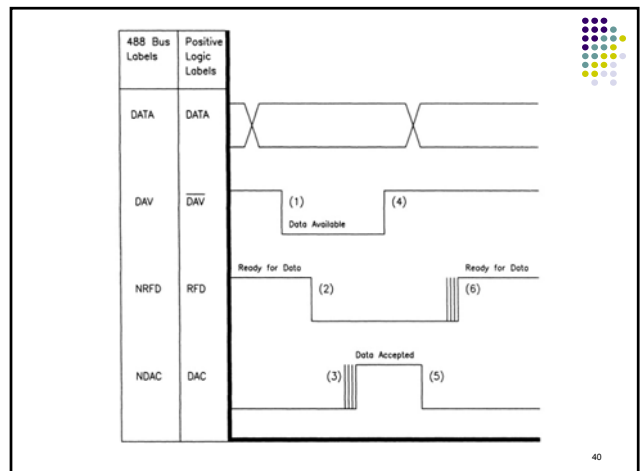


- **Devices:** each device is defined to be:
 - a talker
 - transmits data
 - a listener
 - receives data
 - or a controller
 - sends commands to talkers and listeners and conducts polls to see which devices are active
 - a device can change its function
- **Bus Lines.** These are:
 - 8 bidirectional data lines
 - carry commands, address, and data info
 - 5 general control lines to manage the bus
 - 3 handshaking lines to control each byte transfer
- **Handshaking.** The only part of the standard that will be examined in detail

IEEE-488 Handshaking



- The protocol is fully asynchronous meaning that it does not depend on timing
- Each time a byte is transferred the bus goes through a handshake cycle (see next slide)
- The bus uses negative logic but we will present it in the positive logic
- **DAV^A:** Data available output signal from the talker, active L
- **RFD:** Ready for data output signal from the listener. NRFD means low when not ready for data
- **DAC:** Data accepted output signal from the listener. NDAC means low when data not accepted

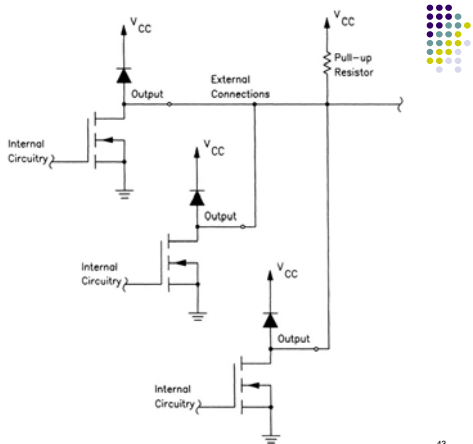


Open-Collector/Open-Drain Outputs

- One solution to the problem of sharing a common wire among gates is to remove the active pull-up transistor from each gate's output
 - for TTL logic => open-collector outputs
 - for CMOS logic => open-drain outputs
- When several open-collector or open drain gates share a common connection the common wire is high by default due to the pull-up
- When one (or more) of the gate outputs pulls it low => the 5V drops across the R_p => common connection is low

Wired-AND/OR Logic With Open-Drain Outputs

- The outputs of an open-drain are a virtual short when the output is low and are high impedance for a logic high
 - the output behaves like a switch
- Since outputs can be connected together the output is high only when all the gate outputs are H
 - the connection is like an AND gate if you think in terms of positive logic
 - If you think in terms of negative logic we see the term wired-OR applied to this type of connection

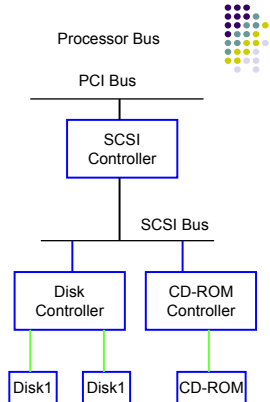


Small Computer System Interface (SCSI)

- This is an interface protocol designed for communications between personal computers and intelligent I/O devices such as disk drives, tape drives, and remote printers
- In its original specification the standard is an ANSI standard: X3.131-186
- SCSI has undergone many revisions, and its data capabilities has increased rapidly:
 - SCSI-1 (obsolete), SCSI-2, 3
- SCSI connector may have 50, 68 or 80 pins
- SCSI bus, data width:
 - 8 data line (narrow bus)
 - 16 data lines (wide bus)
- Electrical signaling allowed:
 - single-ended transmission (SE)- each signal uses one wire with a common ground return for all signals
 - differential signaling – separate return wire is provided for each signal
 - high voltage differential (HVD): 5V TTL
 - low voltage differential (LVD): 3.3 V
- Max. transfer rates up to 320M bytes/s

SCSI

- Devices connected to the SCSI bus are not part of the address space of the processor in the same way as devices connected to the processor bus
- The SCSI bus is connected to the processor bus through a SCSI controller
 - this controller uses DMA to transfer data packets
 - a packet may contain a block of data, commands from the processor to device or status information about the device



SCSI

- To illustrate the mode operation of the SCSI bus, let us consider how it may be used with a disk drive
- Disk to PC communication takes place mainly in bursts of data followed by delays
 - a single read or write request may involve several bursts
- The SCSI is designed to facilitate this mode of operation
- A controller connected to a SCSI bus is of the type
 - an initiator or a target
- An initiator has the ability to select a particular target and to send commands specifying the operations to be performed
 - the controller on the PC side
- The disk controller operates as a target
 - it carries out the commands that it receives

SCSI

- The initiator establishes a logical connection with the intended target
- Once the connection has been established it can be suspended and restored as needed to transfer commands and bursts of data
- While a particular connection is suspended other devices can use the bus
- This ability to overlap data transfer requests is one of the key features of the SCSI bus that leads to high perf.
- Data transfers on the SCSI bus are always controlled by the target controller
 - to send a command to a target, an initiator requests control of the bus
 - after winning arbitration, the initiator selects the target controller and hands control of the bus over to it
 - Then the target controller starts a data transfer to receive a command from the initiator

SCSI

Data block read from a disk

- The processor sends a command to the SCSI controller, which causes the following sequence of events:
 - 1) The SCSI initiator contends for bus control
 - 2) When wins the arbitration => selects the target controller => hand to it the bus control
 - 3) The target starts an output operation (from initiator to target); in response to this the initiator sends a command specifying the required operation
 - 4) The target needs to perform a disk seek op., sends a message to the initiator indicating that will temporarily suspend the connection. Releases bus
 - 5) The target controller sends a command to the disk drive to read data into the buffer. When it is ready the target requests control of the bus. After wins the bus, it reselects the initiator controller, thus restoring the suspended connection

SCSI

Data block read from a disk

- 6) The target transfers the contents of the data buffer to the initiator and then suspends the connection again. Data are transferred either 8 or 16 bits in parallel
- 7) The target controller sends a command to the disk drive to perform another seek operation. Then it transfers the contents on the second disk sector to the initiator, as before. At the end of this transfer, the logical connection is terminated
- 8) As the initiator controller receives the data, it stores them into the main memory using the DMA approach
- 9) The SCSI controller sends an interrupt to the processor to inform it that the requested operation has been completed
- The SCSI bus standard defines a wide range of control messages that can be exchanged between the controllers to handle different types of I/O devices

SCSI

Bus signals

- We show for simplicity the signals for a narrow bus
- Note that all the signal names are preceded by a minus sign – this indicates that the signals are active, or that a data line is equal to 1, when they are in the low-voltage state
- The bus has no address lines
 - instead the data lines are used to identify the bus controllers
 - for a narrow bus there are eight possible controller numbers 0-7
 - a wide bus accommodates 16 controllers
- Activating the corresponding data line is used to indicate the address of a controller

Table 4.4 The SCSI bus signals

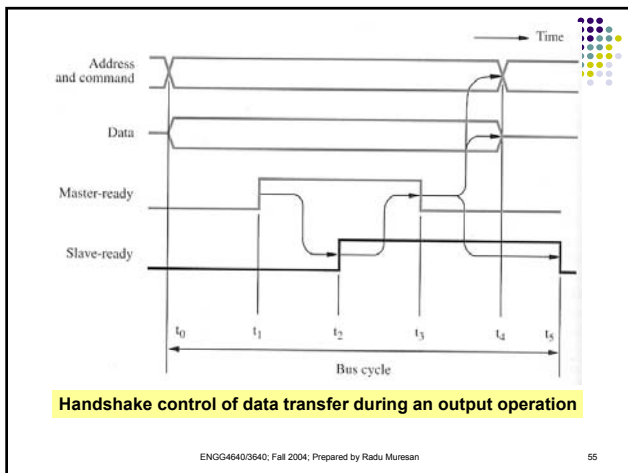
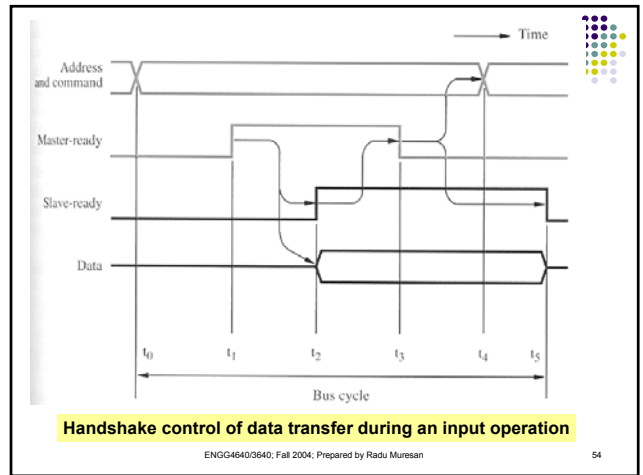
Category	Name	Function
Data	-DB(0) to -DB(7)	Data lines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	-DB(P)	Parity bit for the data bus
Phase	-BSY	Busy: Asserted when the bus is not free
	-SEL	Selection: Asserted during selection and reselection
Information Type	-C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	-MSG	Message: indicates that the information being transferred is a message
Handshake	-REQ	Request: Asserted by a target to request a data transfer cycle
	-ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction transfer	-I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Others	-ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	-RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state

SCSI

- It is possible to have more than one address on the bus at the same time
- The main phases involved in the operation of the SCSI bus are
 - arbitration
 - selection
 - information transfer
 - reselection

Information Transfer

- The information transferred between two controllers can be:
 - commands from the initiator to the target
 - status responses from the target to initiator
 - or data being transferred to or from the I/O
- Handshaking signaling is used to control information transfers
- The -REQ and -ACK signals replace the Master-ready and Slave-ready in an asynchronous handshake bus control
- The target asserts -I/O during an input operation (target to initiator), and it asserts -C/D to indicate that the information being transferred is command or status rather than data



Assignments

- Exercises: Chapter 2, Chapter 8, and Chapter 9 of Spasov book