Introduction to Embedded Systems

EHB326E

Lectures

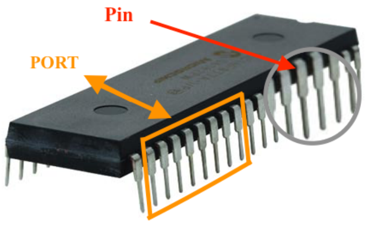
Prof. Dr. Müştak E. Yalçın

Istanbul Technical University

mustak.yalcin@itu.edu.tr

Interface: Ports

- Conducting device on periphery
- Single wire or set of wires with single function
- Connects bus to processor or memory
- Often referred to as a pin
  - Actual pins on periphery of IC package that plug into socket on printed-circuit board
  - Sometimes metallic balls instead of pins
  - Metal "pads" connecting processors and memories within single IC

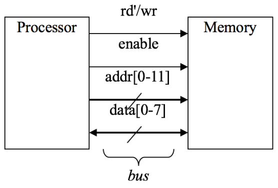


port : a city, town, or other place where ships load or unload.

Interface: Bus

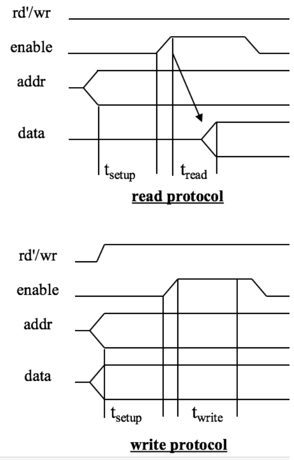
Bus

- Set of wires (Uni-directional or bi-directional) with a single function (Address bus, data bus)
- Or, entire collection of wires (Address, data and control)



Timing Diagrams

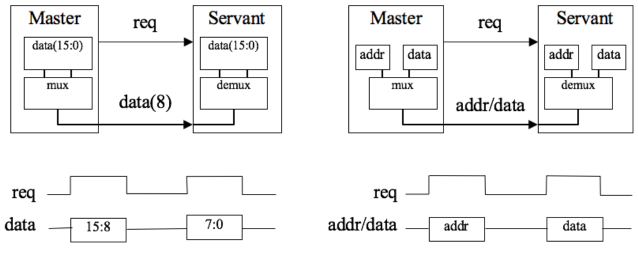
- Most common method for describing a communication protocol
- Time proceeds to the right on x-axis
- Control signal: low or high
  - May be active low (go', $\bar{go}$, or go_L)
  - Use terms assert (active) and deassert
  - Asserting go' means go=0
- Data signal: not valid or valid
- Protocol may have subprotocols
  - Called bus cycle, *e.g.*, read and write
  - Each may be several clock cycles
- Read example
  - rd'/wr set low, address placed on addr for at least t_{setup} time before enable asserted,
  - enable triggers memory to place data on data wires by time t_{read}



Look Figure 1.1 of CMOS EPROM 27C64

Basic Protocol

- Actor: master initiates, servant (slave) respond
- actor can be receiver or sender.
- Addresses: special kind of data used to indicate where regular data should go
- Time multiplexing
  - Share a single set of wires for multiple pieces of data
  - Saves wires at expense of time

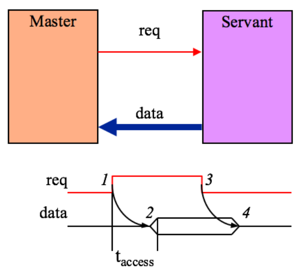


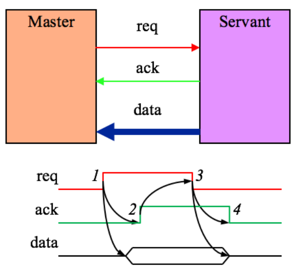
data serializing

address/data muxing

Strobe protocol

Handshake protocol





- Master asserts *req* to receive data
- Servant puts data on bus within time t_{access}
- Master receives data and deasserts *req*
- Servant ready for next request

- Master asserts *req* to receive data
- Servant puts data on bus and asserts *ack*
- Master receives data and deasserts *req*
- Servant ready for next request

- Expansion buses or "slots"
  - ISA - Industry Standard Architecture
  - PCI - Personal Component Interconnect
  - EISA - Extended ISA SIMM - Single Inline Memory Module (plugs into slot) DIMM - Dual Inline Memory Module (plugs into slot)
  - MCA - Micro-Channel Architecture
  - AGP - Accelerated Graphics Port item VESA - Video Electronics Standards Association
  - PCMCIA ? Personal Computer Memory Card International AssociaBon (outdated)
  - PCI-e - PCI express
  - HT - Hypertransport
- Disk interfaces
  - ATA - AT Attachment (named after IBM PC-AT) (outdated)
  - IDE - Integrated Drive Electronics (same as ATA)
  - Enhanced IDE
  - SCSI - Small Computer Systems Interface
  - ESDI - Enhanced Small Device Interface
  - PCMCIA
  - SATA - serial ATA
  - Ethernet (used for network drives)
- External buses
  - Parallel - sometimes called LPT ("line printer")
  - Serial - typically RS232C (RS422)
  - PS/2 - for keyboards and mice
  - USB - Universal Serial Bus
  - IrDA - Infrared Device Attachment
  - FireWire - very high speed, developed by IEEE
- Communications buses
  - Parallel/LPT
  - Serial/RS232C
  - Ethernet

Microprocessor interfacing: I/O addressing

A microprocessor communicates with other devices using some of its pins

- Port-based I/O (parallel I/O)
  - Processor has one or more N-bit ports
  - Processor's software reads and writes a port just like a register

▸ 80C51, page 4-7

- Bus-based I/O
  - Processor has address, data and control ports that form a single bus
  - Communication protocol is built into the processor
  - A single instruction carries out the read or write protocol on the bus

▸ PicoBlaze, page 4-7

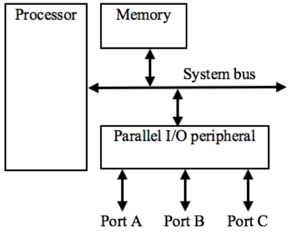
Parallel I/O peripheral

- When processor only supports bus-based I/O but parallel I/O needed
- Each port on peripheral connected to a register within peripheral that is read/written by the processor

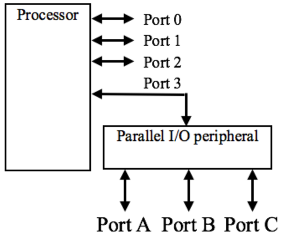
Extended parallel I/O

- When processor supports port-based I/O but more ports needed
- One or more processor ports interface with parallel I/O peripheral extending total number of ports available for I/O

‣ 8255, Programmable Peripheral Interface .



Adding parallel I/O to a bus-based I/O processor



Extended parallel I/O

The ways to talk to peripherals

- Memory-mapped I/O
  - Peripheral registers occupy addresses in same address space as memory
  - e.g., Bus has 16-bit address lower 32K addresses may correspond to memory upper 32k addresses may correspond to peripherals
- Standard I/O (I/O-mapped I/O)
  - Additional pin (M/IO) on bus indicates whether a memory or peripheral access
  - e.g., Bus has 16-bit address all 64K addresses correspond to memory when M/IO set to 0 all 64K addresses correspond to peripherals when M/IO set to 1

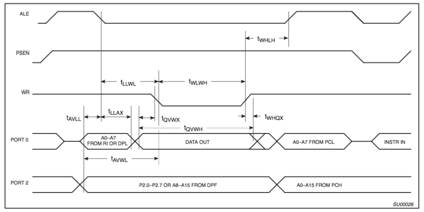
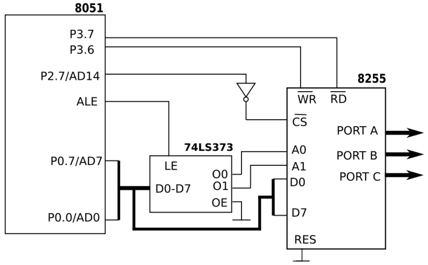
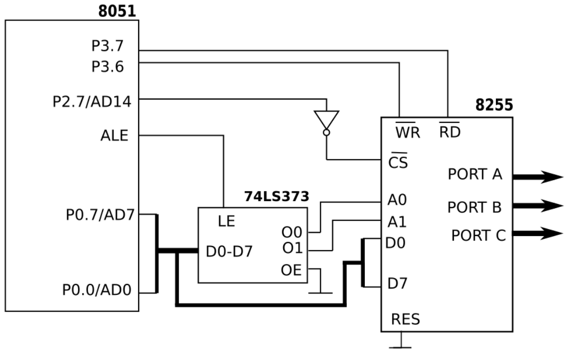


Figure 16. External Data Memory Write Cycle

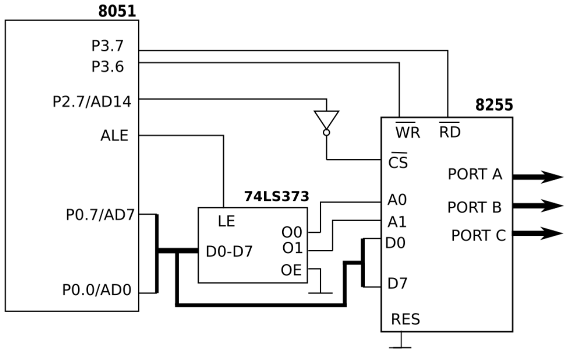
‣ 80C51, page 26



For P3.7 and P3.6 see ‣ 80C51, page 6 and for 74LS373 see ‣ 74LS373



I/O port addresses assigned (memory mapped I/O) to ports A, B, C, and the control register:
x1xxxxxx xxxxxx00=4000h=PORT A
x1xxxxxx xxxxxx01=4001h=PORT B
x1xxxxxx xxxxxx10=4002h=PORT C
x1xxxxxx xxxxxx11=4003h=CONTROL ‣ 8255, page 3-102 .



"All the ports of A, B and C are output ports " then the control word of the 8255 for the configurations (8255, page 3-100) is 10000000b=80h
For P3.7 and P3.6 see 80C51, page 6 and for 74LS373 (Latch) see 74LS373.
A program to send *FFh* to all ports:

$80h \rightarrow @4003h$
$FFh \rightarrow @4000h$
$FFh \rightarrow @4001h$
$FFh \rightarrow @4002h$

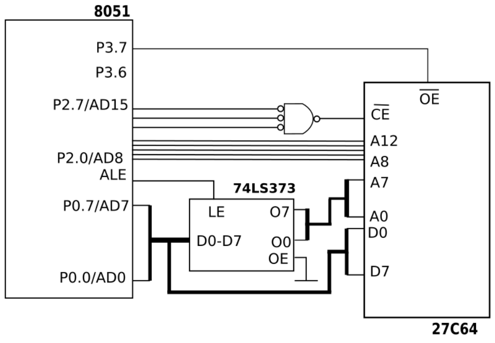
Memory-mapped I/O

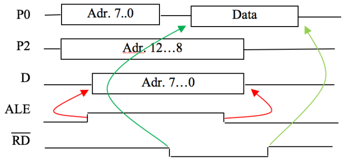
- Requires no special instructions
  - Assembly instructions involving memory like MOV and ADD work with peripherals as well 80C51
  - Standard I/O requires special instructions (INPUT, OUTOUT PicoBlaze Page 32) to move data between peripheral registers and memory

Standard I/O

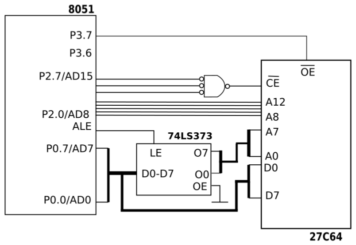
- No loss of memory addresses to peripherals
- Simpler address decoding logic in peripherals possible
  - When number of peripherals much smaller than address space then high-order address bits can be ignored

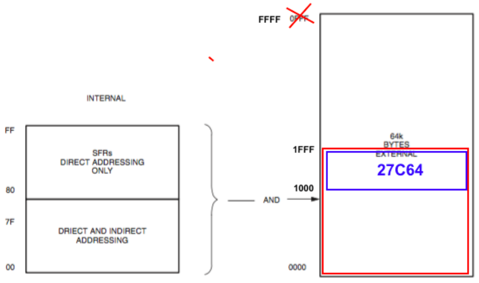
An external ROM uses the 8051 data space to store the data (starting at 1000H). Write a program to read 20 Bytes of these data.





8051, page 6 (P3.7, ALE) CMOS EPROM 27C64 CMOS EPROM 27C64



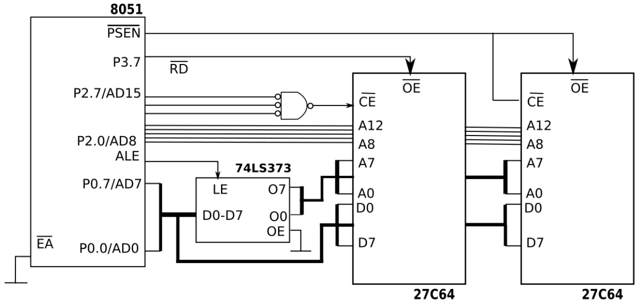


Memory addresses assigned (memory mapped I/O) : 000x xxxx xxxx xxxxb =1FFFh - 0000h 8051, page 6 (P3.7, ALE)

```
        VERI_ADRES EQU 1000h
        KERE EQU 20
        ...
        MOV DPTR, #VERI_ADRES
        MOV R2, #KERE
SUR     MOVX A, @DPTR
        ...
        INC DPTR
        DJNZ R2, SUR
```

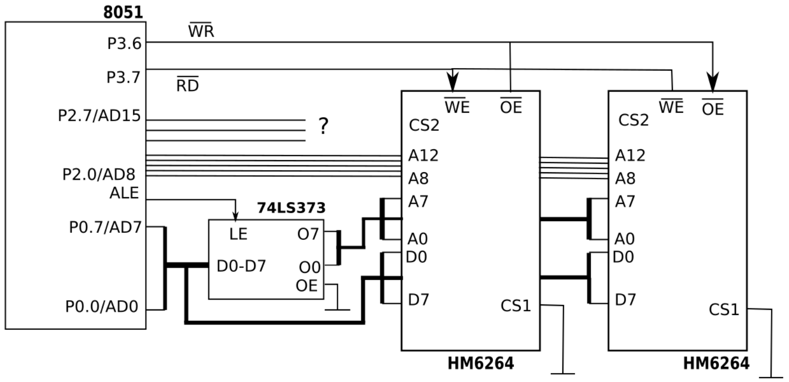
MOVX page 92 8051, page 19 (DPTR), page 25 (timing)

An 8051-based system with 8K bytes of program ROM and 8K bytes of data ROM.



if $\bar{EA} = 1$, PC starts from 0000h(internal memory) then 0FFFh (external memory), if $\bar{EA} = 0$, PC start from external memory (0000h - —-h).

▸ 8051, page 25 (external program memory read cycle)


8051
P3.6
WR
P3.7
RD
P2.7/AD15
?
P2.0/AD8
ALE
P0.7/AD7
P0.0/AD0
74LS373
LE
O7
D0-D7
O0
OE
CS2
WE
OE
A12
A8
A7
A0
D0
D7
CS1
HM6264
CS2
WE
OE
A12
A8
A7
A0
D0
D7
CS1
HM6264