I/O Design

- When designing an I/O port, ensure that the port is only active when selected by the microprocessor
  - Use latches (output) and buffers (input) to isolate the I/O port circuitry from the address and data bus
  - Use the correct combinatorial logic circuitry and/or decoders with address bus to select the port

Input / Output Instructions

- For 8-bit port
  - IN AL, Port #
  - OUT Port #, AL
- For 16-bit port
  - MOV DX, Port #
  - IN AL, DX
  - OUT DX, AL

Example: 8 LEDs

- This is a byte-wide output port
- The LEDs cannot be connected directly to data bus
  - Difficult to select the LEDs
  - LEDs would only display value for very short period of time (about 400ns, or 2 clock cycles)
- Only when data bus carries the correct signal
  - Microprocessor cannot sink enough current

Example: 8 LEDs

- We only want the latch to load values from the data bus when the microprocessor outputs to the correct port #
  - Suggestion 1: Decode the address directly
  - Suggestion 2: Use a decoder such as the 74LS138 with lines from the address bus

Example: 8 LEDs

- Instead, we need to capture the values on the data bus, and hold them until changed
  - The 74LS373 octal latch will do nicely

Example: 8 LEDs

- Note: This is not quite enough!
Example: 8 LEDs

- How do we connect the LEDs?
  - 2 possibilities

Example: 8 LEDs

- We only want the latch to load values from the data bus when the microprocessor outputs to the correct port #
  - Suggestion 1: Decode the address directly
  - Suggestion 2: Use a decoder such as the 3x8 74LS138 with lines from the address bus

Example: 8 LEDs

- Instead, we need to capture the values on the data bus, and hold them until changed
  - The 74LS373 octal latch will do nicely

Note: This is not quite enough!
Example: 8 LEDs

- How do we connect the LEDs?
  - 2 possibilities

**LS138**

The 74LS138 does not have enough power to drive an LED. The device can sink enough current for the LED to light (15 to 20 mA).

Example: 8 LEDs

- Now we will look at an 8-bit input port.
  - The procedure to select the port is similar to the output case
  - Use IORD* instead of IOWR*

Example: 8 Switches

- We cannot use a latch to separate the switches from the microprocessor
  - We only want the switch values to be on the data bus when the microprocessor asks for it
  - A latch would constantly drive the bus!

Example: 8 Switches

- The device of interest here is the 74LS244 tristate buffer (unidirectional)
  - NOT the same as the 74LS245 transceiver (bidirectional)
  - Tristate:
    - One of three states: on (1), off (0), or open (Z)
    - In the open state, the buffer does not drive the data bus

Bus Cycles for outputting

- Assume the port address is 99H
  - OUT 99H, AL
    - T1: address 99H is provided to address bus A0 - A7 through AD0 - AD7 and ALE signal
    - T2: IOW is provided and the contents of AL are released into the data bus pins AD0 - AD7
    - T3: signal propagates to the destination port
    - T4: the content of AL are latched into the 74LS373 with the IOW going from low to high

Example: 8 Switches

- The procedure to select the port is similar to the output case
  - Use IORD* instead of IOWR*
Engr 4862 Microprocessors

Example: 8 Switches

• How do we set up the switches?
  – When open, one logic level
  – When closed, the other logic level

Summary

• Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50-1000ns), it must be latched before it is lost
• In order to prevent any unwanted data from coming into the system data bus, all input devices must be isolated through the tri-state buffer
  - The 74LS244 not only performs this role, but also provides the incoming signals sufficient strength (booster) to travel all the way to the CPU
• As a general rule, every device (memory, peripherals) connected to the global data bus must have a latch or tri-state buffer

Programmable I/O

• The previous examples are good for many applications, but sometimes a more powerful and flexible solution is needed.
  - The 8255 Programmable Peripheral Interface (PPI) is a 40-pin DIP IC that provides 3 programmable I/O ports, A, B, and C.
  - One can program the individual port to be input or output port, economical and flexible than 74LS373, 73LS244, which must be hard wired

Summary

• How are it programmable?
  – Configure each port as input or output
  – Different modes of operation
• You must initialize the PPI via software commands
  - Send a control byte to the device’s control register port

Example: 8 Switches

Engr 4862 Microprocessors

Example: 8 Switches

• How do we set up the switches?
  – When open, one logic level
  – When closed, the other logic level

Programmable I/O

• The previous examples are good for many applications, but sometimes a more powerful and flexible solution is needed.
  - The 8255 Programmable Peripheral Interface (PPI) is a 40-pin DIP IC that provides 3 programmable I/O ports, A, B, and C.
  - One can program the individual port to be input or output port, economical and flexible than 74LS373, 73LS244, which must be hard wired

Summary

• Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50-1000ns), it must be latched before it is lost
• In order to prevent any unwanted data from coming into the system data bus, all input devices must be isolated through the tri-state buffer
  - The 74LS244 not only performs this role, but also provides the incoming signals sufficient strength (booster) to travel all the way to the CPU
• As a general rule, every device (memory, peripherals) connected to the global data bus must have a latch or tri-state buffer

Programmable I/O

• How are it programmable?
  – Configure each port as input or output
  – Different modes of operation
• You must initialize the PPI via software commands
  - Send a control byte to the device’s control register port

Example: 8 Switches

• How do we set up the switches?
  – When open, one logic level
  – When closed, the other logic level

Programmable I/O

• The previous examples are good for many applications, but sometimes a more powerful and flexible solution is needed.
  - The 8255 Programmable Peripheral Interface (PPI) is a 40-pin DIP IC that provides 3 programmable I/O ports, A, B, and C.
  - One can program the individual port to be input or output port, economical and flexible than 74LS373, 73LS244, which must be hard wired

Summary

• Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50-1000ns), it must be latched before it is lost
• In order to prevent any unwanted data from coming into the system data bus, all input devices must be isolated through the tri-state buffer
  - The 74LS244 not only performs this role, but also provides the incoming signals sufficient strength (booster) to travel all the way to the CPU
• As a general rule, every device (memory, peripherals) connected to the global data bus must have a latch or tri-state buffer

Programmable I/O

• How are it programmable?
  – Configure each port as input or output
  – Different modes of operation
• You must initialize the PPI via software commands
  - Send a control byte to the device’s control register port

Example: 8 Switches

• How do we set up the switches?
  – When open, one logic level
  – When closed, the other logic level

Programmable I/O

• The previous examples are good for many applications, but sometimes a more powerful and flexible solution is needed.
  - The 8255 Programmable Peripheral Interface (PPI) is a 40-pin DIP IC that provides 3 programmable I/O ports, A, B, and C.
  - One can program the individual port to be input or output port, economical and flexible than 74LS373, 73LS244, which must be hard wired

Summary

• Since the data provided by the CPU to the port is on the system data bus for a limited amount of time (50-1000ns), it must be latched before it is lost
• In order to prevent any unwanted data from coming into the system data bus, all input devices must be isolated through the tri-state buffer
  - The 74LS244 not only performs this role, but also provides the incoming signals sufficient strength (booster) to travel all the way to the CPU
• As a general rule, every device (memory, peripherals) connected to the global data bus must have a latch or tri-state buffer

Programmable I/O

• How are it programmable?
  – Configure each port as input or output
  – Different modes of operation
• You must initialize the PPI via software commands
  - Send a control byte to the device’s control register port

Example: 8 Switches

Pin Description

• PA0 — PA7: Port A / All / input/output/bidirectional
• PB0 — PB7: Port B / All / input/output
• PC0 — PC7: Port C / All / input/output
  - Can be split into two parts: Upper (PC7 — PC4) and Lower (PC3 — PC0)
  - Each can be used for input or output.
  - Any of PC0 — PC7 can be programmed.
• RD and WR: control signal input to 8255
  - RD and WR in peripheral I/O
  - MEMR and MEMW in memory-mapped I/O
Engr 4862 Microprocessors

**Pin Description**

- **RESET**: Active high input signal to 8255
  - Used to clear the internal control register
  - When activated, all ports are initialized as input ports
  - Usually connected to the RESET output of the system bus or ground
- **A0, A1, and CS**
  - CS selects the entire chip, A0 and A1 select the specified port or control register
  - Use to access port A, B, C

<table>
<thead>
<tr>
<th>Mode</th>
<th>Select</th>
<th>Port A</th>
<th>Port B</th>
<th>Port C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Port A</td>
<td>0</td>
<td>Port B</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Port C</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Control Reg.</td>
<td>0</td>
<td>Port C</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Not Selected</td>
<td>1</td>
<td>Port C</td>
</tr>
</tbody>
</table>

**Mode Selection**

- **Mode 0**: simple I/O
  - Any ports: A, B, CL, CU. No control of individual bits
- **Mode 1**: I/O (ports A and B) with handshaking (port C)
  - Synchronizes communication between an intelligent device (printer)
- **Mode 2**: Bidirectional I/O with handshaking
  - Port A: Bidirectional I/O with handshaking through port C
  - Port B: Simple I/O or in handshake mode 1
- **BSR Mode**: Bit set/reset
  - Only the individual bits on Port C can be programmed

**Control Word of 8255**

- **Output 0**
  - Output 0: 0 = read, 1 = write
  - Output 1: 0 = 0, 1 = not 0

- **Output A**
  - Output 0: 0 = read, 1 = write
  - Output 1: 0 = 0, 1 = not 0

**8255 Design Example**

- **Mode 0**: Any of ports A, B, C can be programmed as input or output
  - Port cannot be both an input and output port at the same time
  - Port C can be programmed with CL, CH separately
  - Example:

**Mode Selection**

- **It's the control register that must be programmed to select the operation mode of the three ports: A, B, and C**
- **The 8255 chip is programmed in any of the above modes by sending a byte (control word) to the control register of the 8255**

**8255 Design Example**

- **Input 0**
  - Input 0: 0 = read, 1 = write
  - Input 1: 0 = 0, 1 = not 0

- **Input A**
  - Input 0: 0 = read, 1 = write
  - Input 1: 0 = 0, 1 = not 0

- **Input CL**
  - Input 0: 0 = read, 1 = write
  - Input 1: 0 = 0, 1 = not 0

- **Input CH**
  - Input 0: 0 = read, 1 = write
  - Input 1: 0 = 0, 1 = not 0