Parallel I/O

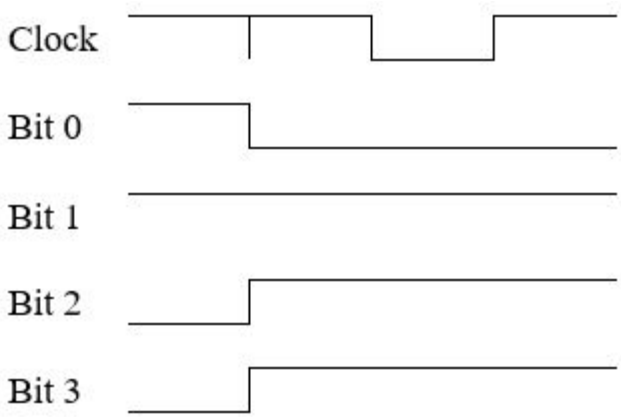
All together now

Parallel I/O

- An n-bit value can be transmitted in parallel, meaning all bits simultaneously over n lines.
  - The alternative is to transmit serially
- As with serial communication
  - A set of communication lines is referred to as a “bus”
  - Parallel communication can be synchronous or asynchronous
- For synchronous, data transfer operates as illustrated on the next slide
  - Some additional details related to synchronous busses may be discussed later in the course as time permits

Parallel I/O

- E.g. illustrate transmission of 4-bit value assuming a shared clock signal

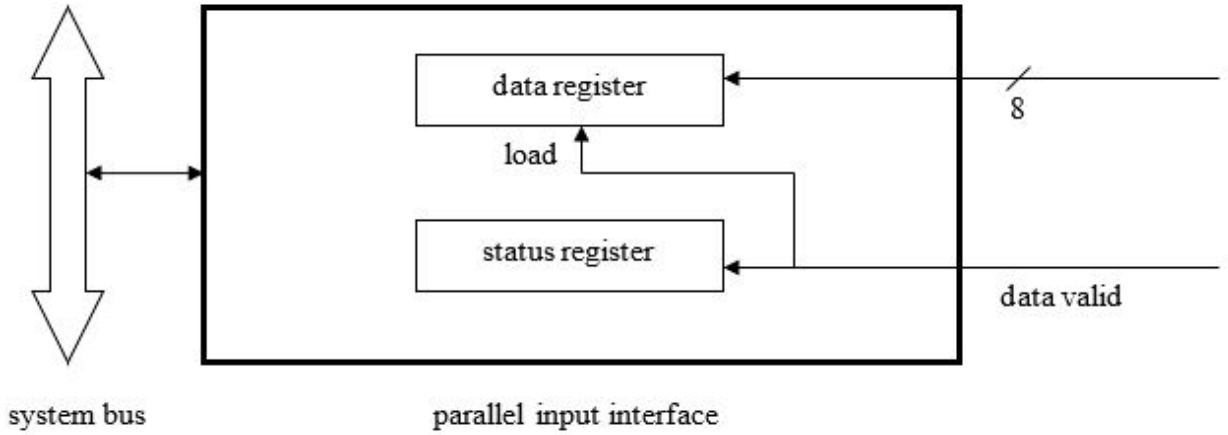


Asynchronous Parallel I/O

- How does an asynchronous parallel I/O interface know when incoming data is valid and ready to be latched?
- In the serial situation THE data line is held in a known state and there is a special bit that is different from this state so the transition is detectable.
- In this case there are multiple data lines and the signals on each of them can be different.
- It makes no sense to require a special value to proceed each data value.
- Thus, a special signal, a “data valid” signal, must be supplied to the receiver.

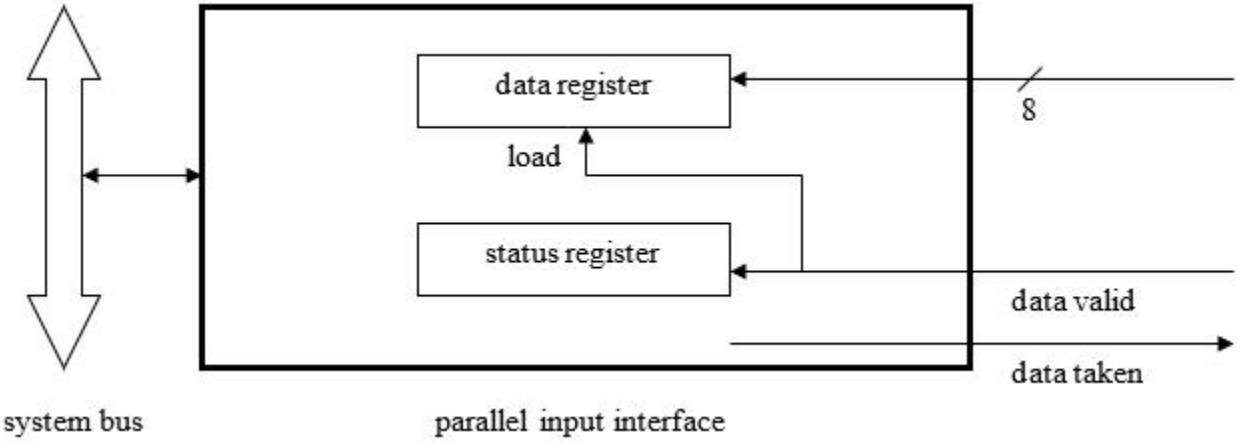
Asynchronous Parallel I/O

- This diagram shows how a data valid line is used to manage incoming data
- Why does the data valid line also go to the status register?



Asynchronous Parallel I/O

- How does the parallel interface stop new data from overwriting the previous data value, if the CPU hasn't read the value yet?
- It must supply a "data taken" signal.

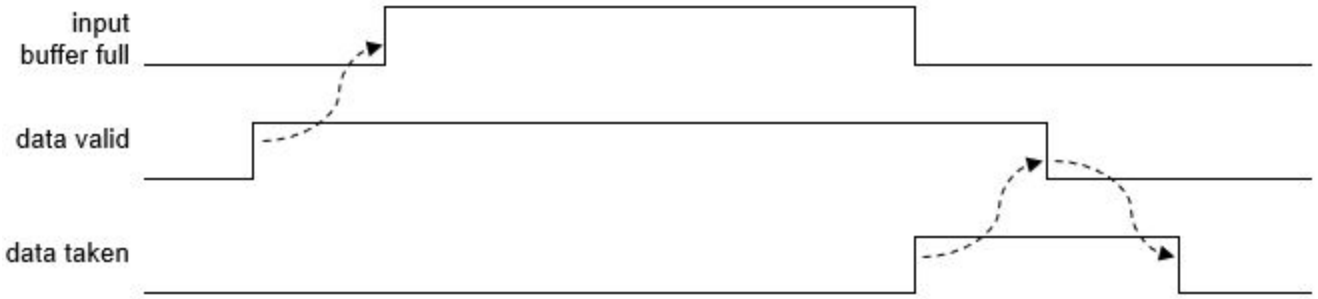


Asynchronous Parallel I/O

- The “data taken” signal is generated by the interface’s control logic (not shown) whenever valid data in the data register is read by the CPU
- Thus, the communication between sender and receiver can be synchronized
    - Still remain “asynchronous” in the sense that there is no shared clock
- These control signals implement a “handshaking” protocol.

Asynchronous Parallel I/O

- The following is a “timing diagram” which illustrates a full handshake:
  - At what point in the diagram is the data register read by the CPU?



Asynchronous Parallel I/O

- Where the input buffer full bit falls and the data taken signal is asserted
- There are also partial (or “pulse mode”) handshakes, in which the control signals are pulsed instead of left high until acknowledged
- Would a handshaking protocol ever be needed for serial communications?
    - Yes, but not for the same purpose. For serial, handshaking can be used for flow control and it can be implemented either using dedicated control lines (out of band) or as part of the data stream (in band – maybe as a software protocol).
    - Example: TCP/IP Networking Protocol

Asynchronous Parallel I/O Interfaces

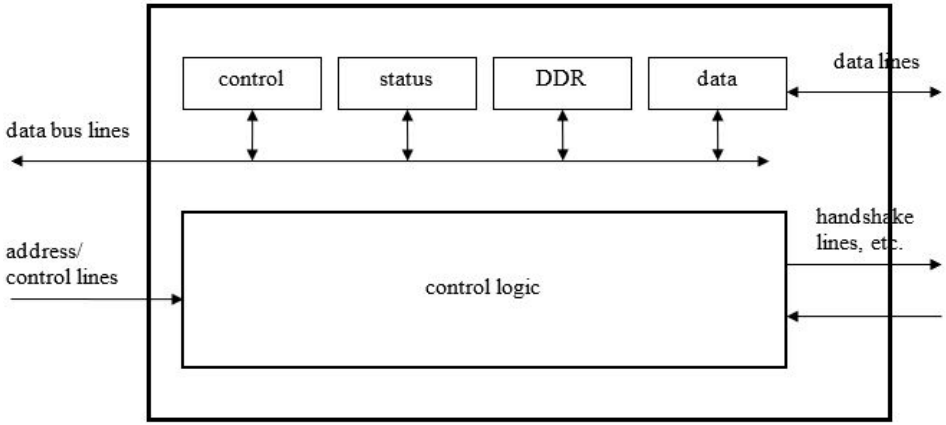
- Typical CPU-accessible registers:
  - Command
  - Status
  - Data
  - Data direction register (DDR)
- If a DDR is present, each of its bits specifies the direction of individual port data lines (and hence bits in the data register). Thus, it may be possible to use a parallel interface for simultaneous bidirectional communication

Asynchronous Parallel I/O Interfaces

- With DDR:
  - Bit == 0 means input
  - Bit == 1 means output
  - E.g. with 8 lines total, 5 could be for input and 3 could be for output.
  - 5 input buttons on a vending machine
  - 3 outputs for item selected / coin return (transaction cancelled)

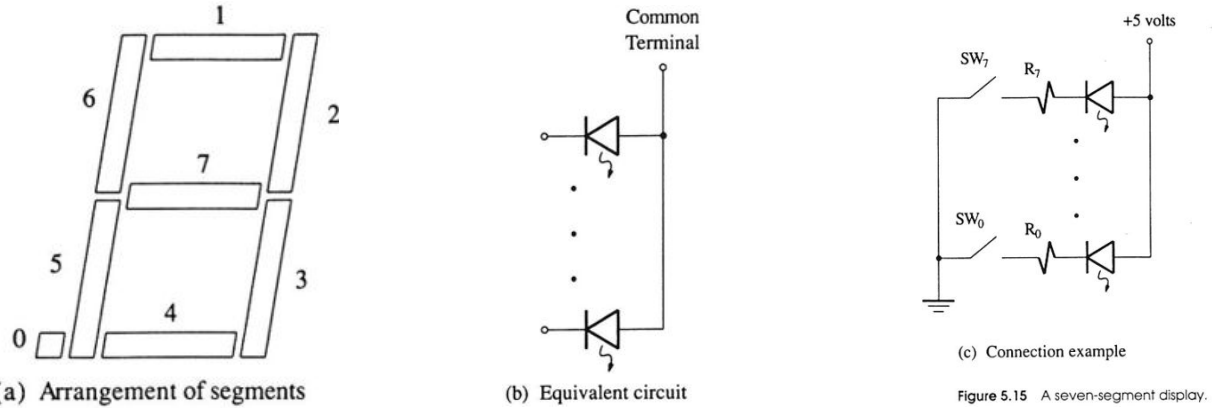
Asynchronous Parallel I/O Interfaces

- Generalized organization and external signals:
- Typical control bits:
  - Interrupt enable
  - Handshake mode select (e.g. full vs. pulse-mode)
- Typical status bits:
  - Data valid
  - Interrupt requested



Asynchronous Parallel I/O

- E.g. on the Atari ST, the parallel port (usually used for connecting a printer) is controlled by the YM2149 PSG!
- Note that a parallel interface can be used for a variety of I/O purposes.
- Example: A 7 segment display



(a) Arrangement of segments

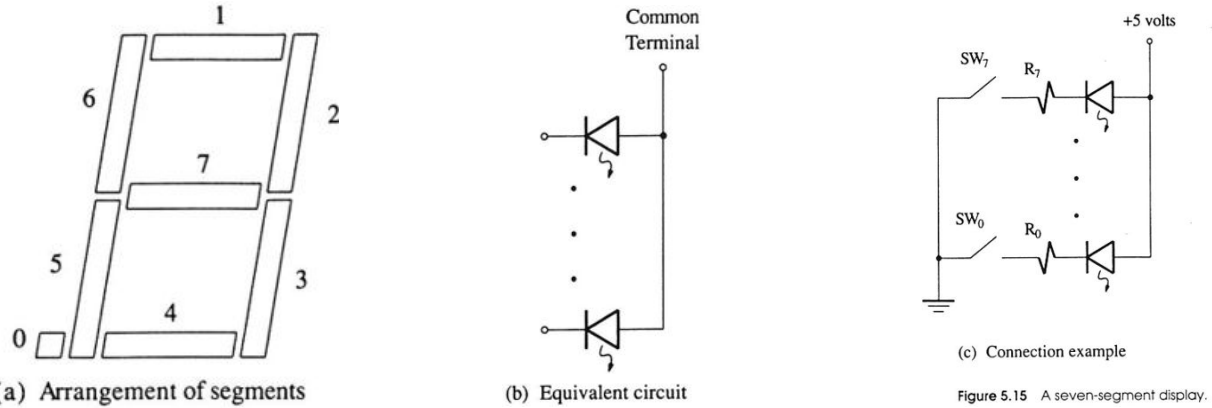
(b) Equivalent circuit

(c) Connection example

Figure 5.15 A seven-segment display.

Asynchronous Parallel I/O

- E.g. on the Atari ST, the parallel port (usually used for connecting a printer) is controlled by the YM2149 PSG!
- Note that a parallel interface can be used for a variety of I/O purposes.
- Example: A 7 segment display



(a) Arrangement of segments

(b) Equivalent circuit

(c) Connection example

Figure 5.15 A seven-segment display.

Asynchronous Parallel I/O

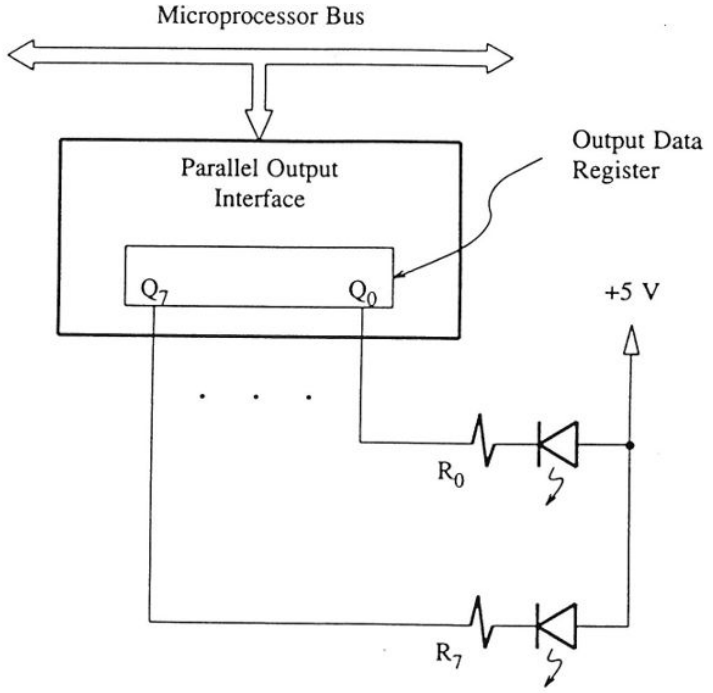


Figure 5.16 A seven-segment display (8 LEDs) driven by a parallel interface circuit.

Asynchronous Parallel I/O

- Example: Reading an array of switches
  - Microcomputer Structures, Z.G. Vranesic & S.G. Zaky, Holt, Rinehart & Winston, 1989
  - In both cases the values are inverted: 1 means OFF and 0 means ON

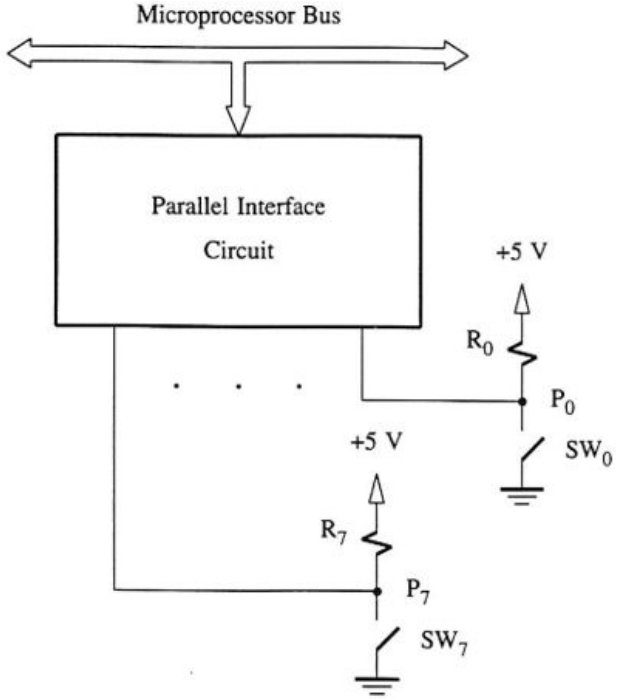


Figure 5.14 A parallel interface connecting 8 switches to a microprocessor.

IEEE 1284 Port

- IEEE 1284 is a standard (followed to some degree) for parallel (“Centronics”) ports. In the past, these ports were commonly used for connecting a printer, although USB has mostly replaced them
- A 36-pin version was superseded by a 25-pin version. The remaining pins are ground pins.
- Signals include:
  - Data (× 8)
  - Strobe (“data valid”)
  - Ack (“data taken”)
  - Initialize
  - Auto-feed
  - Online
  - Busy
  - Fault
  - Paper empty

Serial vs. Parallel

- Parallel busses are very common (e.g. the system bus which connects the CPU to memory)
- All else being equal, parallel is faster than serial, in theory
- However, serial busses such as USB are increasingly used for connecting peripherals
    - Why?

Serial vs. Parallel

- Cost
  - Electrical considerations
- Cons of Serial
  - In theory, slower than parallel (all else being equal)

Serial vs. Parallel

- Pros of Serial
  - The time required to transmit in parallel is the same, whereas, the time required to transfer 1 bit has decreased significantly
  - In practice, sometimes faster than parallel because it can be clocked at a higher frequency (no skew, little or no crosstalk)
  - Narrower, cheaper and more robust cabling, esp. for longer distances (compare USB cables to 25-pin parallel “ribbon” cables)
  - Smaller connectors (imagine a 25-pin parallel connector on an iPod nano)