EE414 Embedded Systems

Ch 4. Standard Single Purpose Processors: Peripherals

Part 2/5: Parallel Interface

Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

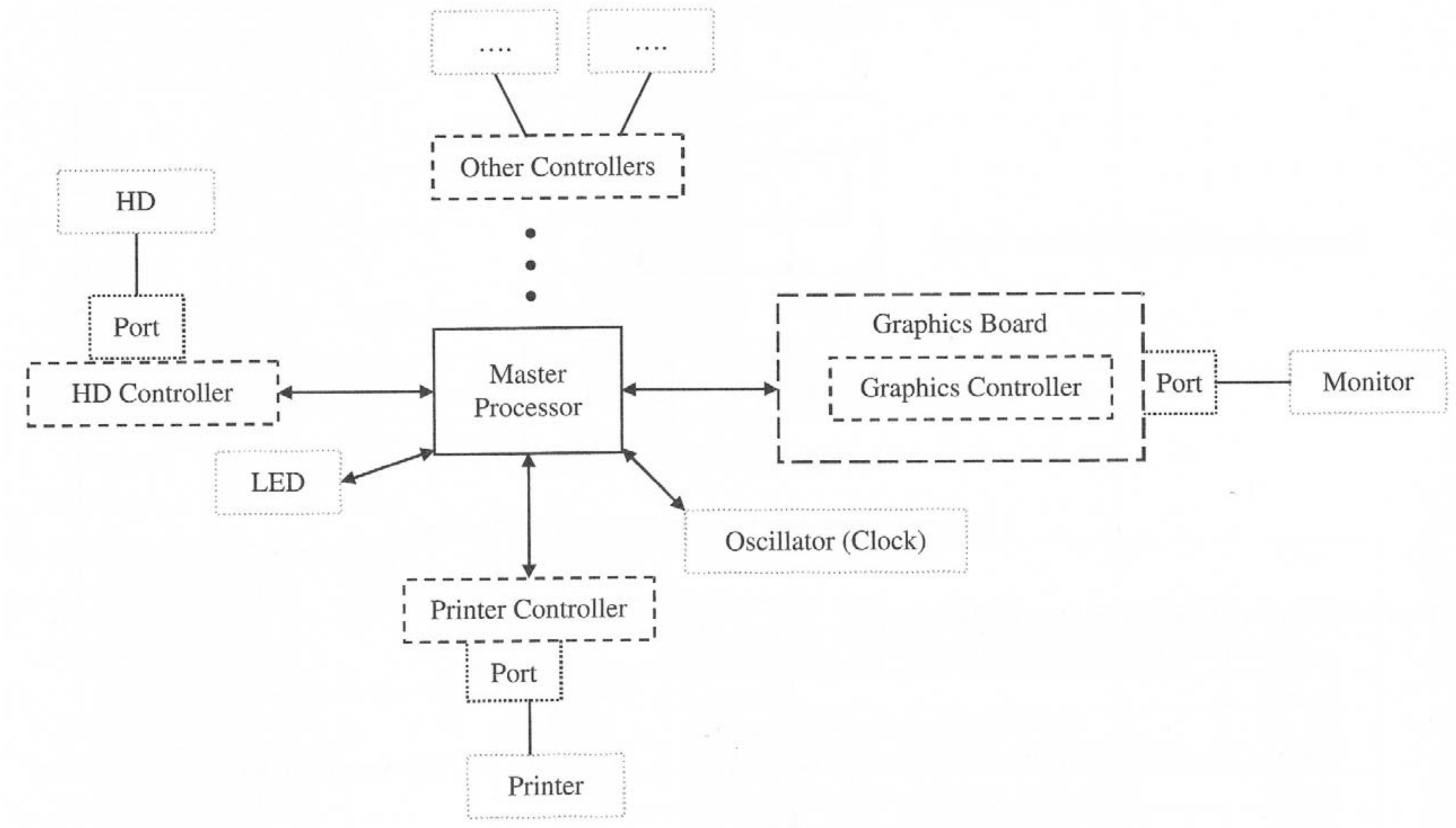
Overview

Ch 4. Standard Single Purpose Processors: Peripherals

Part 2/5. Parallel Interface

- 4.11 Introduction to Input/Output
- 4.12 Input/Output Mechanisms
- 4.13 Centronix Interface
- 4.14 IEEE 1284

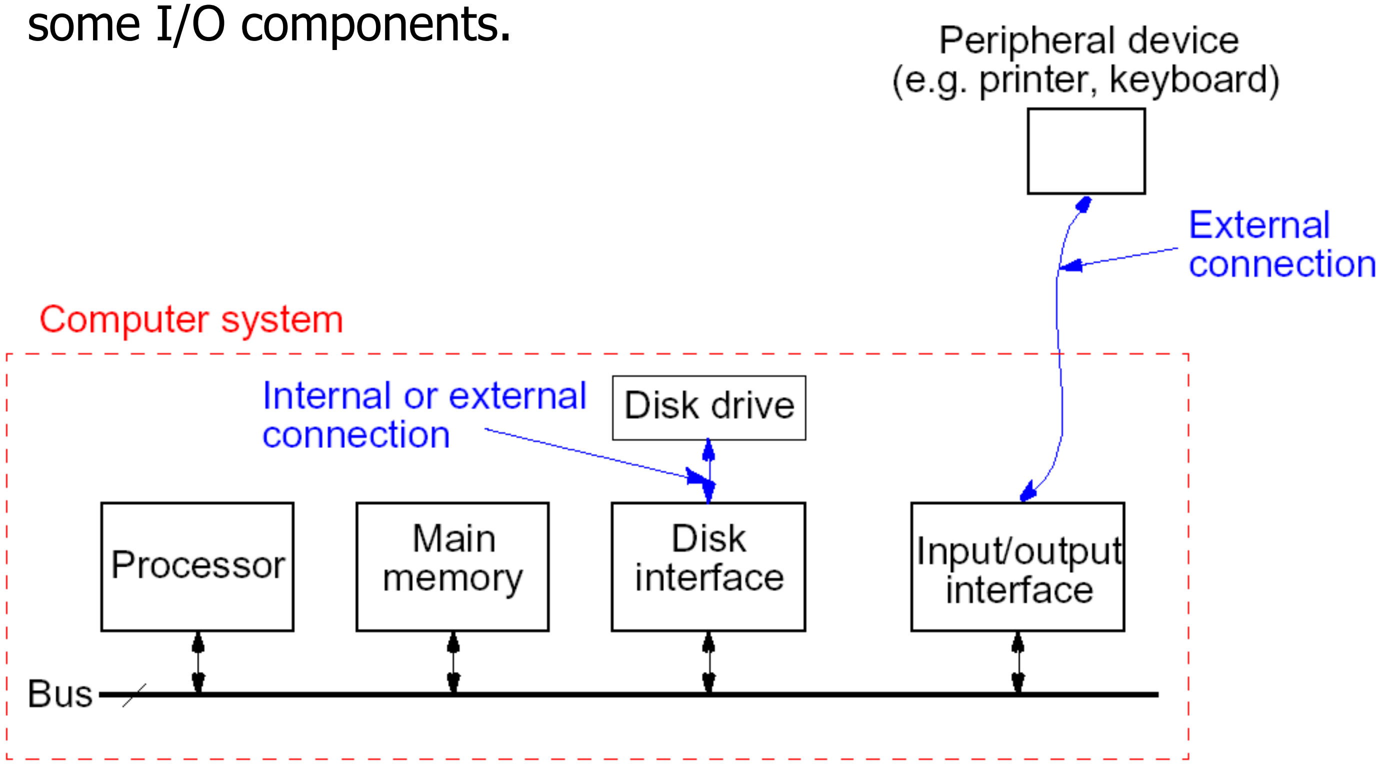
4.11 Introduction to Input/Output



- Num wires?

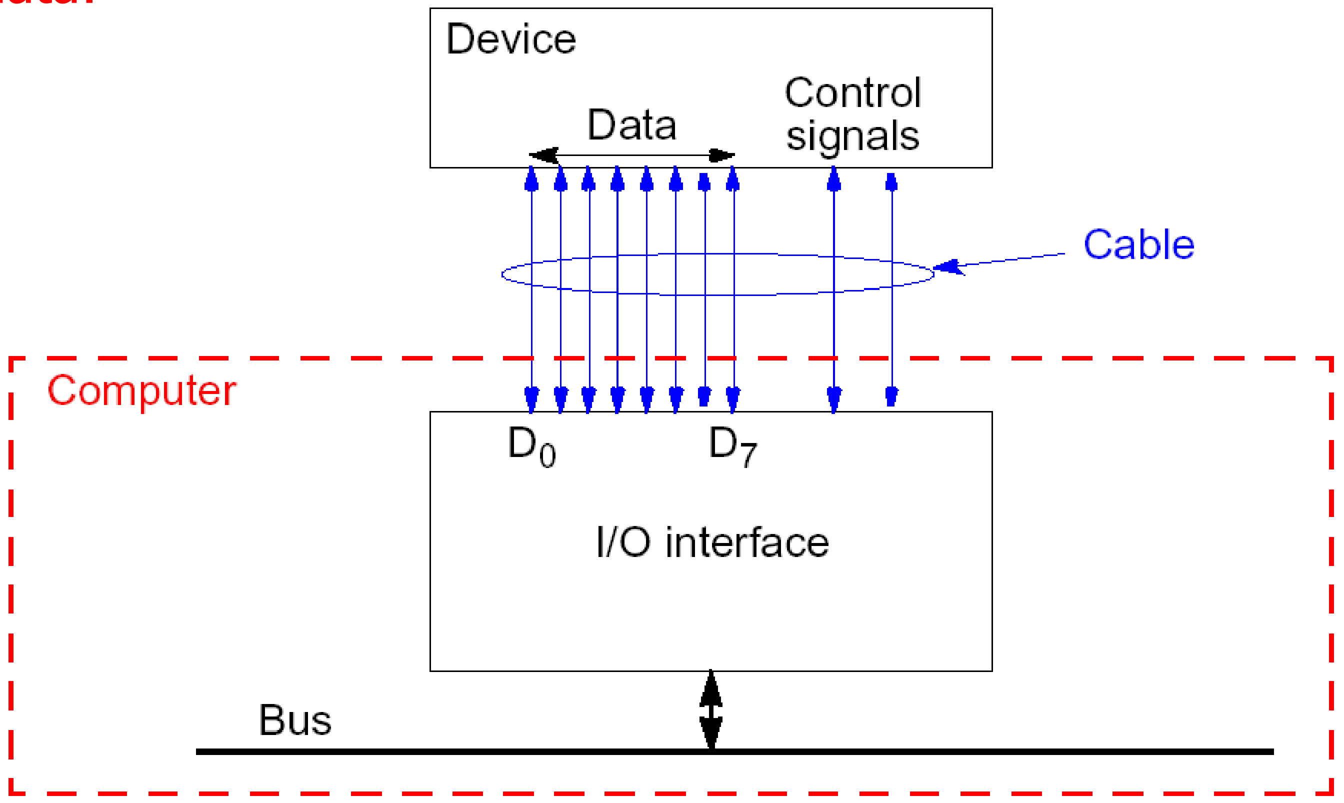
Introduction to Input/Output (II)

- **Input/Output**
  - System with one bus connecting processor, memory, and I/O interfaces.
    - Note: Modern systems have separate buses for main memory and some I/O components.



Basic Parallel Interface

- **Parallel Interface**
  - Data is transmitted with one wire assigned to each bit of the data:



4.12 Input/Output Mechanisms

- **Input/Output Mechanisms**
  - Instructions and mechanisms must be present
    - to enable data to be brought into the computer from an external device
    - and for data to be transferred to an external device.
  - Each type of device has different characteristics and may requires different techniques. Here we will outline:

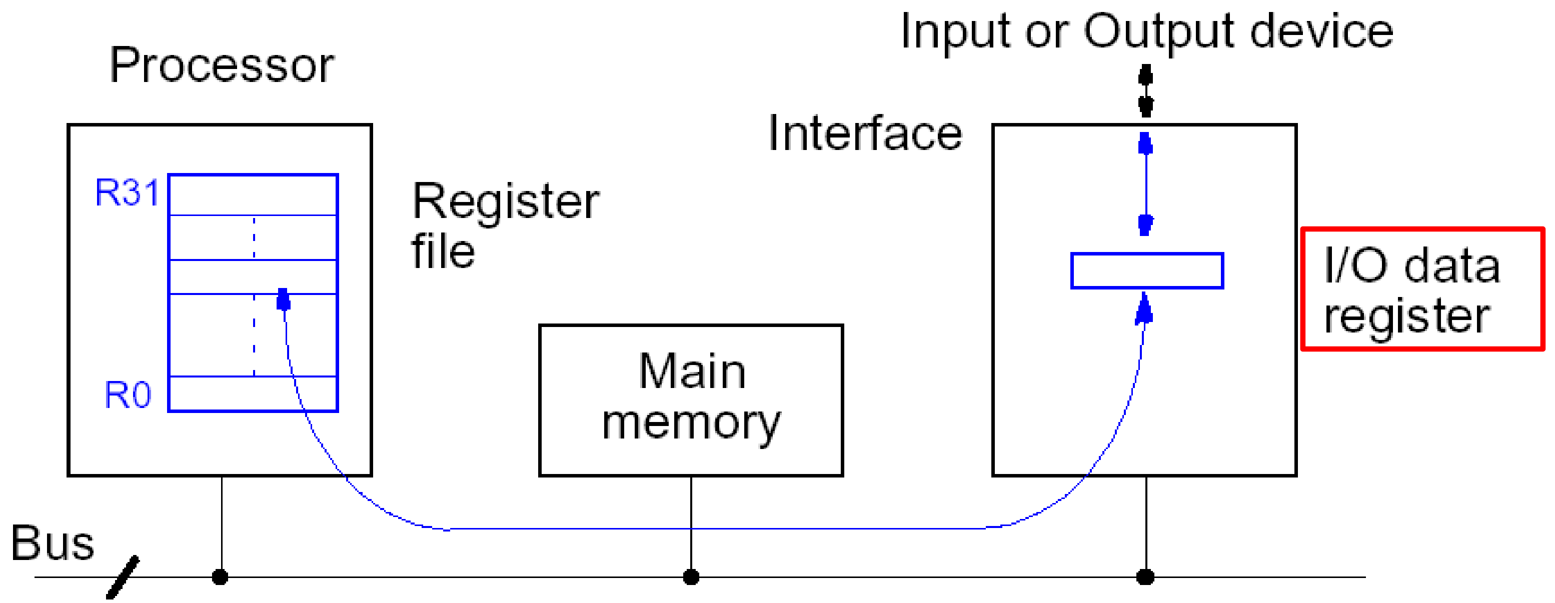
    1. Programmed Input/Output (in detail)

    2. Interrupt mechanism

    3. Direct memory access

Input/Output Mechanisms (II)

- **Programmed Input/Output: I/O Data Register**
  - Machine instruction provided for transferring data to and from I/O device.
  - Transfer instigated by the instruction is between addressable I/O registers within the interface and processor registers.

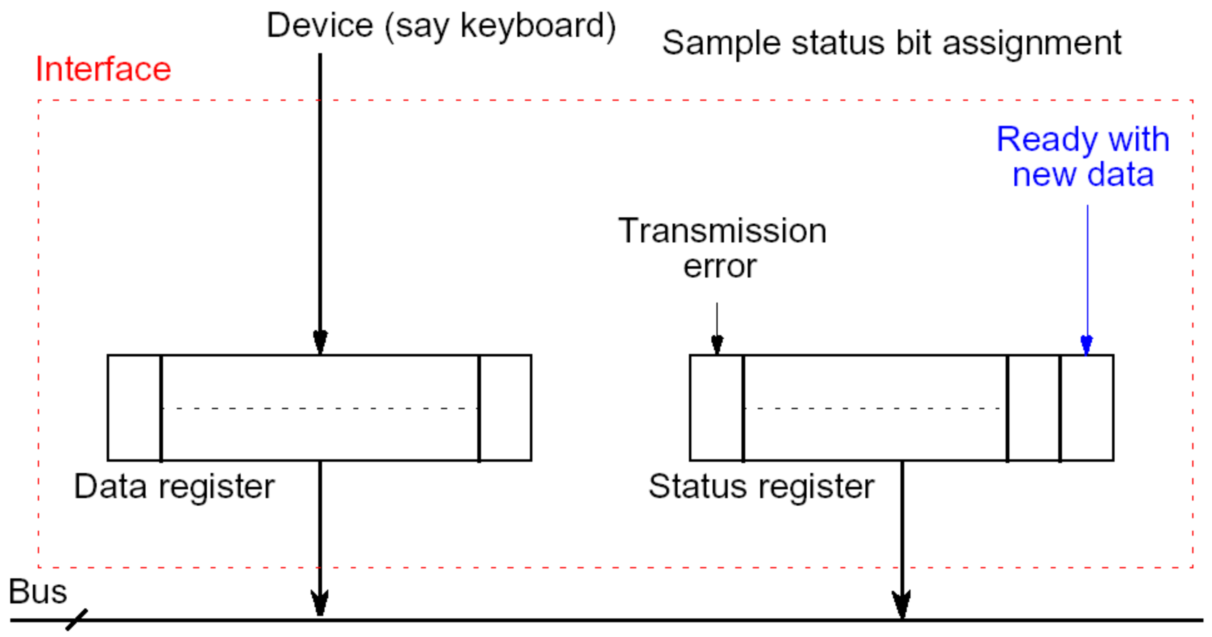


Input/Output Mechanisms (III)

- **I/O Status Registers**
  - Programmed I/O usually require programmer to check:
  - Output: interface is ready to accept a new data.
    - CPU writes → Interface accepts → Device output → interface is ready to accept
  - Input: interface is ready with new data to read or
    - Device input → interface is ready with new data to read → CPU reads
  - Done by reading an interface I/O status register prior to making each data transfer (polling).
  - Individual bits in the status register can indicate:
    1. Whether interface is ready to accept a new data.
    2. Whether interface has new data received from device to read.
    3. Whether any error has occurred in transmission to/from device.

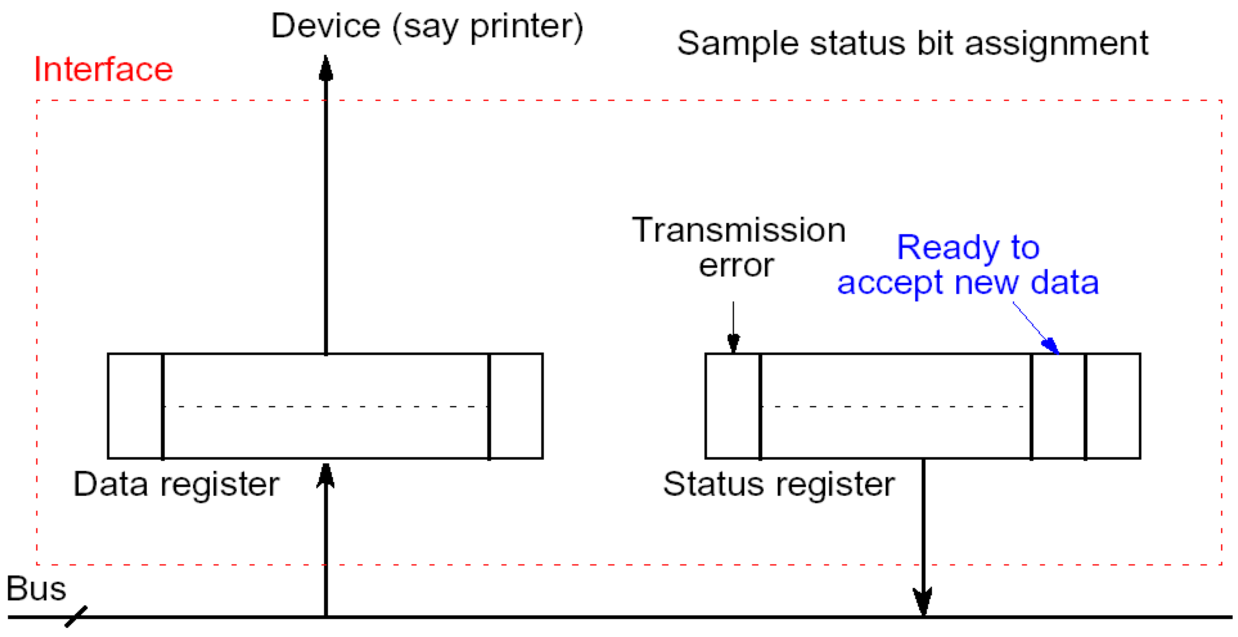
Input/Output Mechanisms (IV)

- **Example Input Interface**
  - 0) Human input keyboard
  - 1) Device set "Data reg" and "Ready with new data" bit *set* when new data loaded by device.
  - 2) CPU checks status: "Ready with new data" bit (iteratively)
  - 3) If ready, CPU reads data: Device "Ready with new data" bit *reset* when Data reg taken (read) by processor.



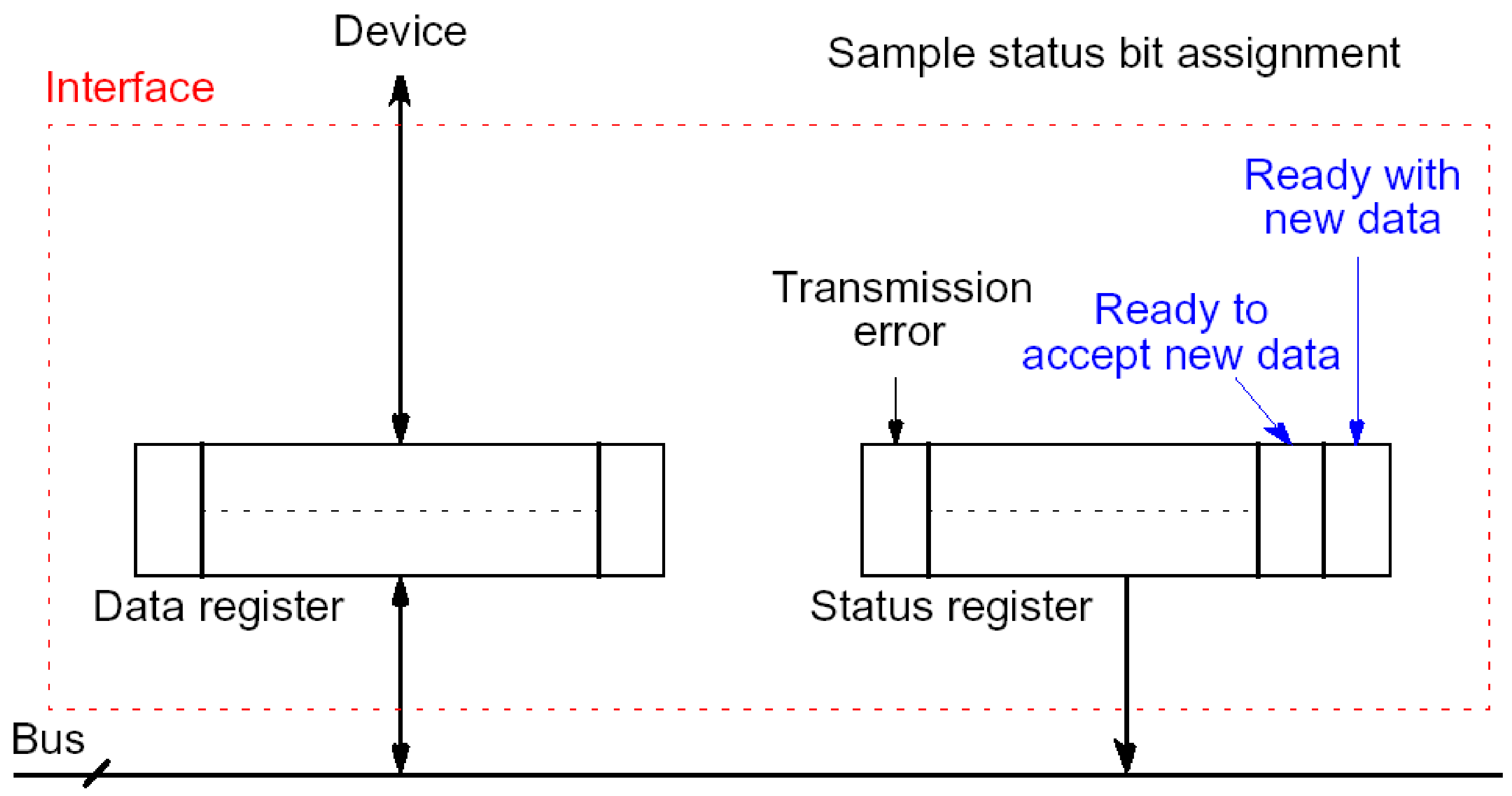
Input/Output Mechanisms (V)

- **Example Output Interface**
  - 0) CPU check status: “Ready to accept new data” (iteratively)
  - 1) CPU writes Data reg. Device *reset* “Ready to accept new data” bit when new data loaded by processor.
  - 2) Device processes Data
  - 3) Device *set* “Ready to accept new data” bit when done (e.g. printed).



Input/Output Mechanisms (VI)

- **Bidirectional interface**



Input/Output Mechanisms (VII)

- **I/O addressing**
  - Each register within the I/O interface is given an address:
    - 1. Memory mapped
      - memory address not used in main memory
    - 2. I/O mapped
      - from a separate I/O address space.
- *I/O instructions*
  - Memory mapped I/O registers enable normal memory references instructions (LD/ST) to be used to access I/O registers.
  - I/O mapped I/O registers require special IN/OUT instructions.

Input/Output Mechanisms (VIII)

- **Programmed Input/Output With Memory Mapped I/O Registers**
  - Suppose
    1. Address of status register held in R3
       1. (Bit 0 = 1 if data is ready. Bit 1 = 1 accept is ready)
    2. Address of data input register address held in R4,
    3. Address of data output register address held in R5, and
    4. Bits of status register assigned as previously.
  - To read/write data from/to interface (or device)

```
L1:     LD R1,[R3]          ;read status register
        AND R1,R1,001B
        BZ L1               ;go back if data not ready
        LD R2,[R4]          ;get data and put in R2
```

```
L2:     LD R1,[R3]          ;read status register
        AND R1,R1,010B
        BZ L2               ;go back if data not ready
        ST [R5],R2          ;send data held in R2 If not ready
```

Input/Output Mechanisms (IX)

- **Programmed Input/Output with I/O Mapped I/O registers**
  - Typically the op-code mnemonics are **IN** and **OUT**.
  - Examples:
    ```
    IN R1,34          ;Copy contents of I/O register 34
                      ;into processor register R1
    OUT 45,R5         ;Copy contents of processor
                      ;register R5 into I/O register 45
    ```
  - Polling sequence similar to previously except use **IN** and **OUT** instead of **LD** and **ST**.
  - Generally memory mapped I/O preferred even if processor supports I/O mapped instructions.

Input/Output Mechanisms (X)

- **Errors in Transmission**
  - Connection from interface to device may be vulnerable to interference and errors may occur.
  - To combat this, an error detection (and sometimes correction) mechanism usually in place.
- **Parity bit method**
  - A simple and most common method is the parity bit method.
  - Extra bit called a parity bit attached to data word, which is set to either a 0 or to a 1 so as to make the total number of 1's in the complete word (including the parity bit) even - for "even" parity.
  - Then, complete word sent. At receiving end, number of 1's checked. If not even, an error must have occurred.
  - Example

    Data word = **10010101**

    Add partity bit = **0**

    Complete word = **010010101**

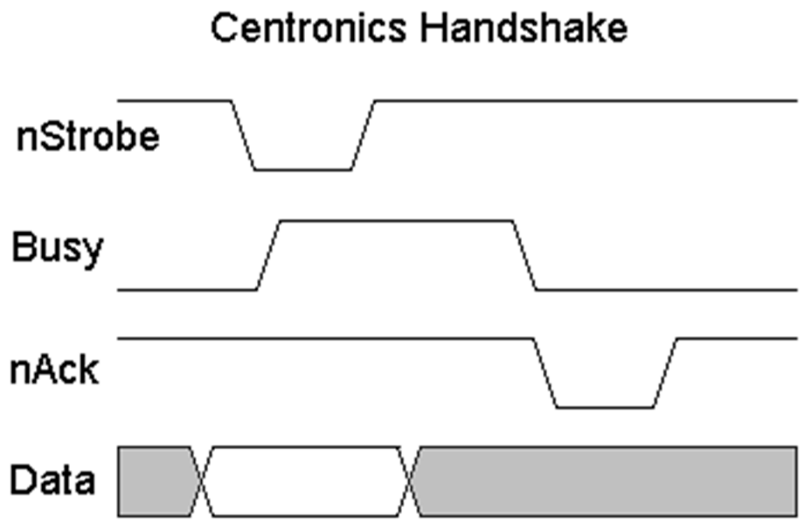
    Suppose pattern received is = **010010111**
    - This has an odd number of 1's so must be an error!

Input/Output Mechanisms (XI)

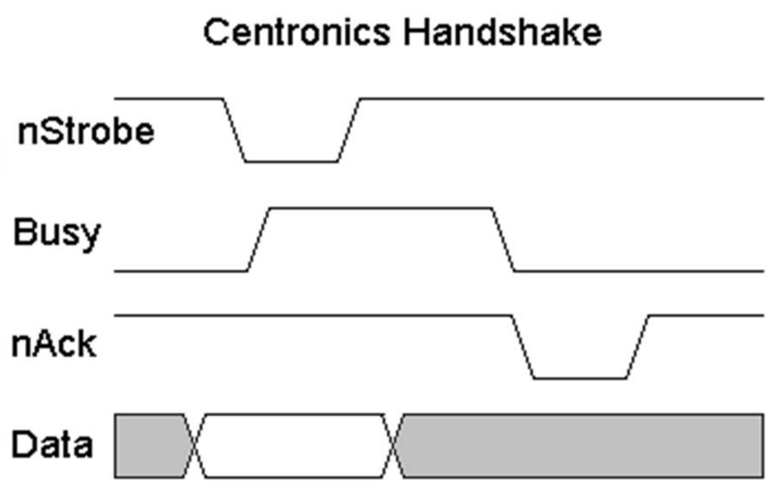
- Polling approach (Programmed I/O)
  - Only suitable for slow systems that can wait between transfers.
  - Unsuitable:
    - 1. When the time of new input data is indeterminate, e.g. waiting for someone to press control-*C* to stop a program.
    - 2. When the time between new data is very short, e.g. successive data bytes from/to a hard disk drive.
- Solution
  - For 1, we use the "*interrupt mechanism*"
  - For 2, we use "*direct memory access*"

4.13 Centronix Interface

- **Centronics**
  - An early standard for transferring data from a host to the printer.
  - The majority of printers used this handshake.
  - This handshake is normally implemented using a Standard Parallel Port under software control.
  - Simplified diagram of the `Centronics' Protocol:



Centronix Interface (II)



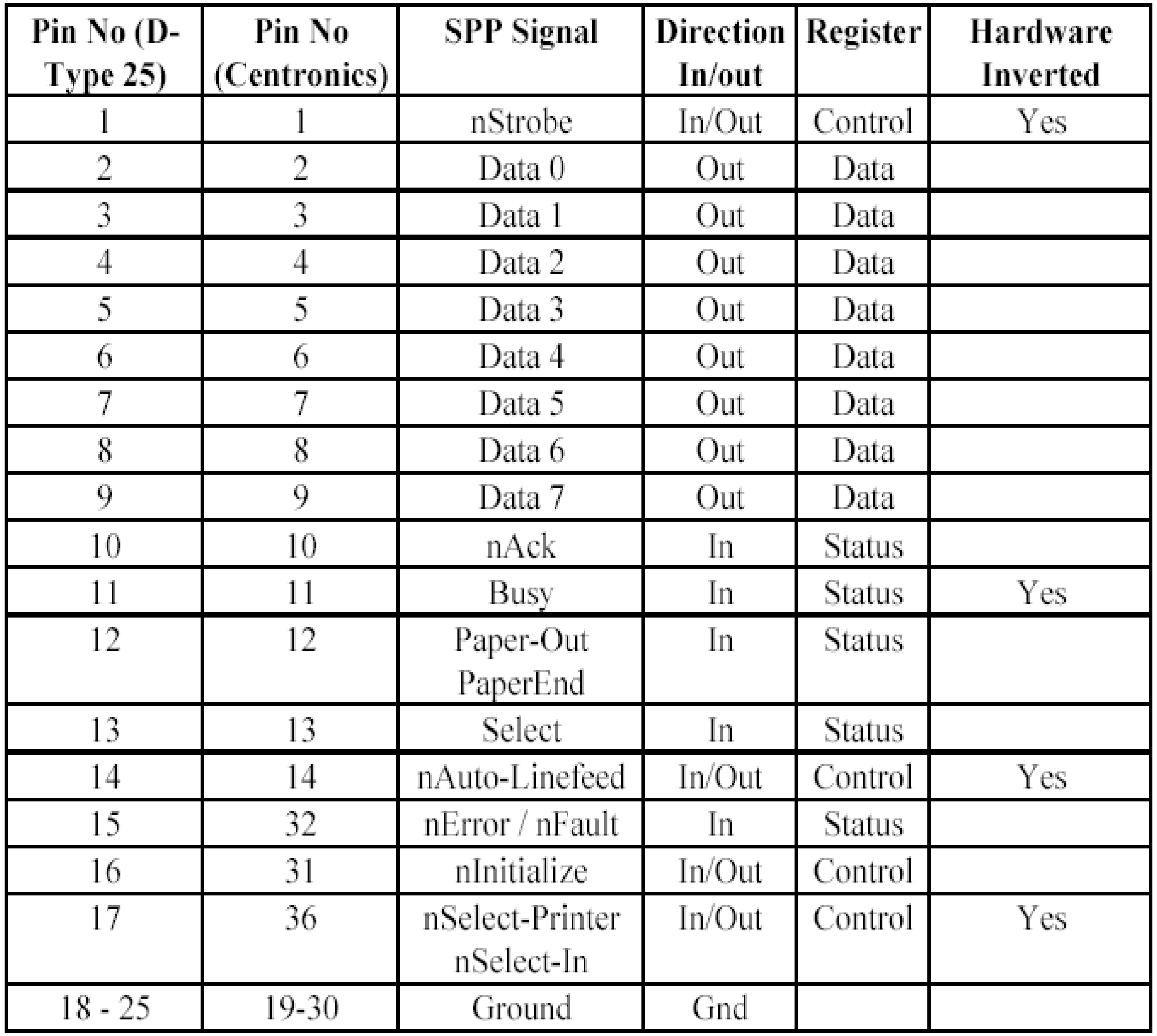
- **Centronics operation**
  - 1) Data is first applied on the Parallel Port pins 2 to 9.
  - The host then checks to see if the printer is busy. i.e. the busy line should be low.
  - 2) The program then asserts the strobe, waits a minimum of 1uS, and then de-asserts the strobe.
  - Data is normally read by the printer/peripheral on the rising edge of the strobe.
  - 3) The printer will indicate that it is busy processing data via the Busy line.
  - 4) Once the printer has accepted data, it will acknowledge the byte by a negative pulse about 5uS on the nAck line.

Centronix Interface (III)

- **Centronix pin-outs**
  - D-Type 25 Pin connector (IEEE 1284 Type A)
    - The most common connector found on the Parallel Port of the computer
  - Centronics 34 Pin connector (IEEE 1284 Type B)
    - Commonly found on printers. The IEEE 1284 standard however specifies 3 different connectors for use with the Parallel Port. The first one, is the D-Type 25 connector found on the back of most computers. The 2nd is the which is the 36 pin Centronics Connector found on most printers.
  - IEEE 1284 Type C
    - A 36 conductor connector like the Centronics, but smaller.
    - Claimed to have a better clip latch, better electrical properties and is easier to assemble.
    - Contains two more pins for signals which can be used to see whether the other device connected, has power.
    - Recommended for new designs.

Centronix Interface (IV)

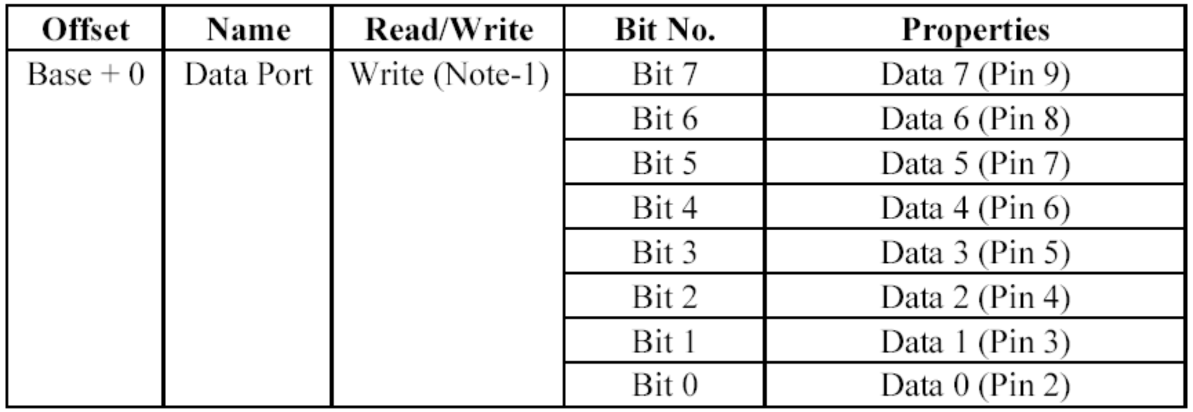
- **Centronix pin-outs**



Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	nStrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	nAck	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out PaperEnd	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	nInitialize	In/Out	Control	
17	36	nSelect-Printer nSelect-In	In/Out	Control	Yes
18 - 25	19-30	Ground	Gnd		

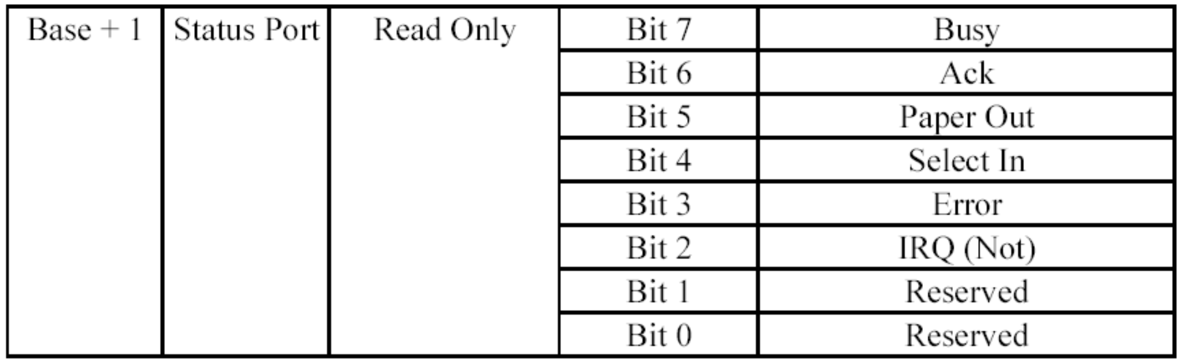
Centronix Interface (V)

- Port addresses and registers for IBM PC
  - 378h-37Fh: LPT1
  - 278h-27Fh: LPT2



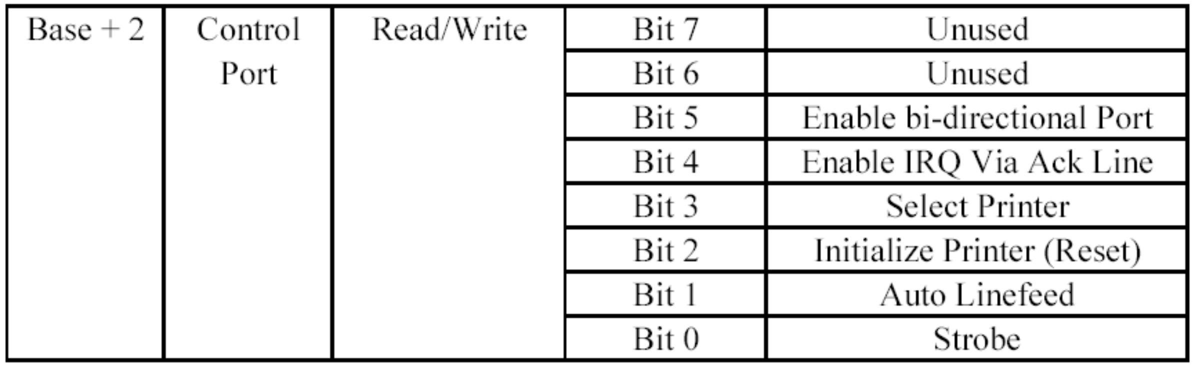
Offset	Name	Read/Write	Bit No.	Properties
Base + 0	Data Port	Write (Note-1)	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

Table 4 Data Port



Base + 1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

Table 5 Status Port

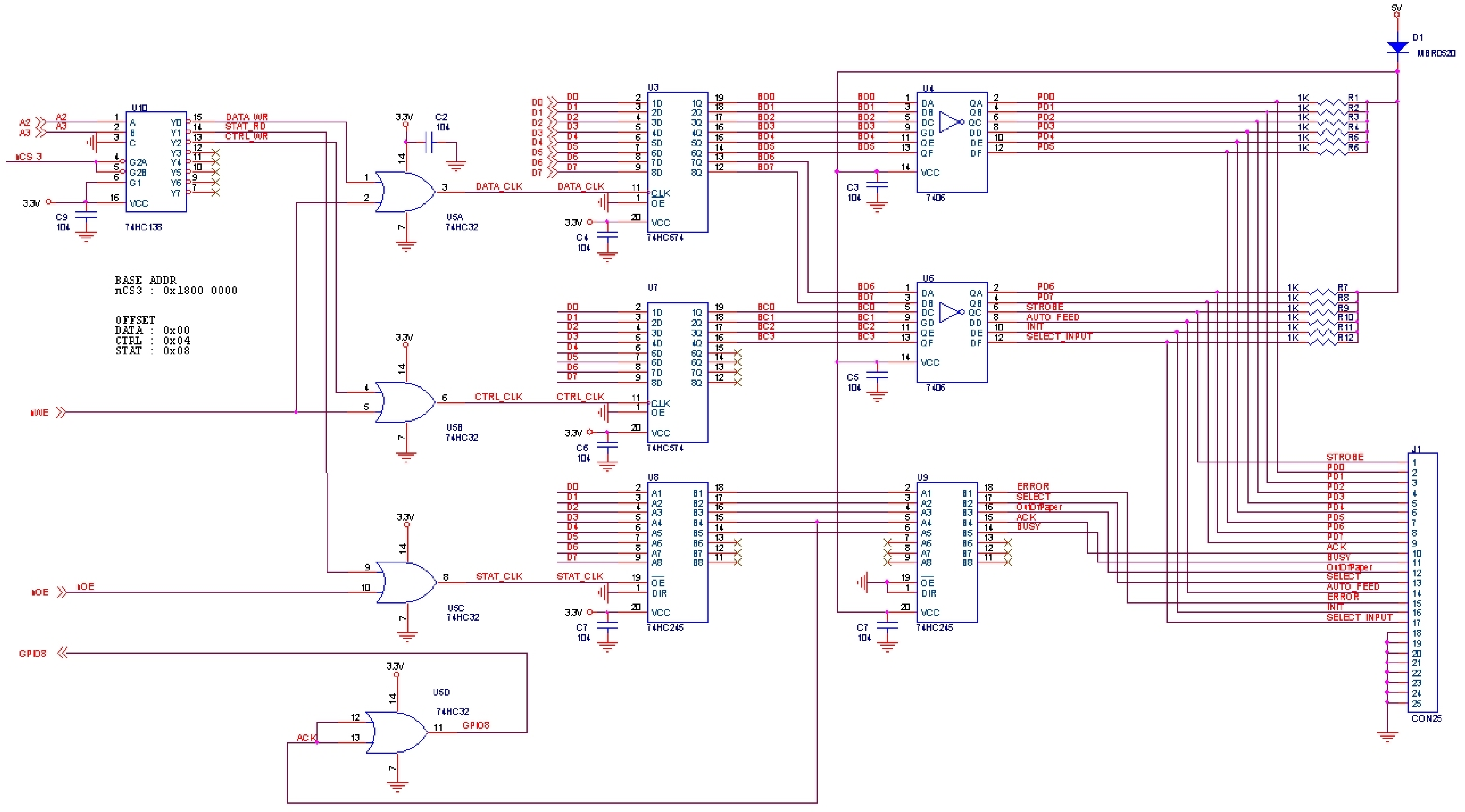


Base + 2	Control Port	Read/Write	Bit 7	Unused
			Bit 6	Unused
			Bit 5	Enable bi-directional Port
			Bit 4	Enable IRQ Via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Table 6 Control Port

Centronix Interface (VI)

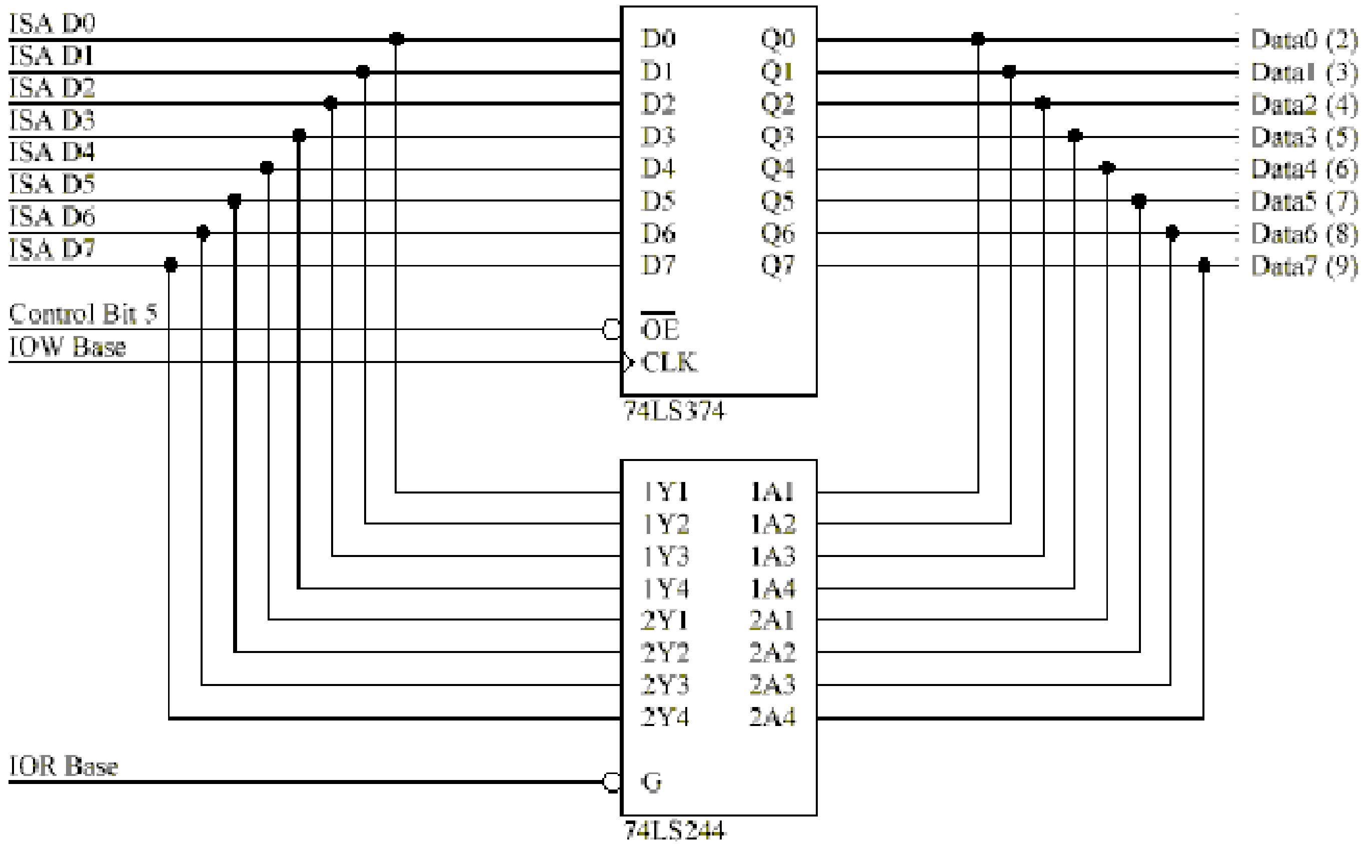
Interface Circuit for EzBoard



Centronix Interface (VII)

- Bi-directional port implementation

Standard Parallel Port Bi-Directional Operation



4.14 IEEE 1284

- Newer Parallel Ports are standardized under **the IEEE 1284 standard** first released in 1994.
- This standard defines 5 modes of operation which are as follows:

  1. Compatibility Mode.
  2. Nibble Mode.
  3. Byte Mode.
  4. EPP Mode (Enhanced Parallel Port).
  5. ECP Mode (Extended Capabilities Mode).

- **Aim of IEEE 1284**
  - To design new drivers and devices which were compatible with each other and also backwards compatible with the Standard Parallel Port (SPP).

IEEE 1284 (II)

- **1. Compatibility mode or "Centronics Mode"**
  - Can only send data in the forward direction at a typical speed of 50 kbytes per second but can be as high as 150+ kbytes a second.
- **2. Nibble and Byte mode**
  - In order to receive data, you must change the mode to either Nibble or Byte mode.
    - Nibble mode can input a nibble (4 bits) in the reverse direction. E.g. from device to computer.
    - Byte mode uses the Parallel's bi-directional feature (found only on some cards) to input a byte (8 bits) of data in the reverse direction.

IEEE 1284 (IV)

- **3. Enhanced (EPP) and Extended Parallel Ports (ECP)**
  - Use additional hardware to generate and manage handshaking.
  - The EPP & ECP ports speed up by
    - Letting the hardware check to see if the printer is busy and generate a strobe and /or appropriate handshaking.
    - Only one I/O instruction need to be performed, thus increasing the speed.
    - Can output at around 1-2 megabytes per second.
    - The ECP port also has the advantage of using DMA channels and FIFO buffers, thus data can be shifted around without using I/O instructions.

References

- Parallel interface, Input output mechanism, IEEE 1284
  - Search Internet
- Centronix Interface in PC
  - IBM, "Technical Reference - Personal Computer XT Hardware Reference Library", 1986
- Centronix interface
  - http://www.falinux.com