Hardware-Software Interfaces
CSEE W4840

Prof. Stephen A. Edwards

Columbia University

Spring 2013
Processor System Block Diagram

Processor

Address Bus

Data Bus

Memory

Peripheral

Peripheral
Simple Bus Timing

Read Cycle

Write Cycle
Strobe vs. Handshake

**Strobe**
- Req
- Data

**Handshake**
- Req
- Ack
- Data
1982: The IBM PC/XT
The ISA Bus: Memory Read
The ISA Bus: Memory Write

![Diagram showing the timing of a memory write operation on the ISA bus. The diagram includes signals for Clk, Addr, BALE, MEMR, IOCHRDY, and Data. The timing is divided into phases C1, C2, Wait, C3, and C4. Each phase shows the rise and fall of the signals.](image-url)
The PC/104 Form Factor: ISA Lives

Embedded System Legos. Stack ‘em and go.
Memory-Mapped I/O

- To a processor, everything is memory.
- Peripherals appear as magical memory locations.
- Status registers: when read, report state of peripheral
- Control registers: when written, change state of peripheral
Typical Peripheral: PC Parallel Port

At Standard TTL Levels

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Adapter Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strobe</td>
<td>1</td>
</tr>
<tr>
<td>+Data Bit 0</td>
<td>2</td>
</tr>
<tr>
<td>+Data Bit 1</td>
<td>3</td>
</tr>
<tr>
<td>+Data Bit 2</td>
<td>4</td>
</tr>
<tr>
<td>+Data Bit 3</td>
<td>5</td>
</tr>
<tr>
<td>+Data Bit 4</td>
<td>6</td>
</tr>
<tr>
<td>+Data Bit 5</td>
<td>7</td>
</tr>
<tr>
<td>+Data Bit 6</td>
<td>8</td>
</tr>
<tr>
<td>+Data Bit 7</td>
<td>9</td>
</tr>
<tr>
<td>Ack</td>
<td>10</td>
</tr>
<tr>
<td>+Busy</td>
<td>11</td>
</tr>
<tr>
<td>+Paper End</td>
<td>12</td>
</tr>
<tr>
<td>+Select</td>
<td>13</td>
</tr>
<tr>
<td>-Auto Feed</td>
<td>14</td>
</tr>
<tr>
<td>-Error</td>
<td>15</td>
</tr>
<tr>
<td>-Initialize</td>
<td>16</td>
</tr>
<tr>
<td>-Select Input</td>
<td>17</td>
</tr>
<tr>
<td>Ground</td>
<td>18-25</td>
</tr>
</tbody>
</table>

Pin allocations:
- pin 1
- pin 13
- pin 14
- pin 25

Timing diagrams:
- Strobe
- Busy
- Ack
- Data
Parallel Port Registers

<table>
<thead>
<tr>
<th></th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x378 Busy</td>
<td>Ack</td>
<td>Paper</td>
<td>Sel</td>
<td>Err</td>
<td>Sel</td>
<td>Init</td>
<td>Auto</td>
<td>Strobe</td>
</tr>
<tr>
<td>0x379 Sel</td>
<td>Init</td>
<td>Auto</td>
<td>Strobe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Write Data
2. Assert Strobe
3. Wait for Busy to clear
4. Wait for Acknowledge
A Parallel Port Driver

```c
#define DATA    0x378
#define STATUS  0x379
#define CONTROL 0x37A

#define NBSY    0x80
#define NACK    0x40
#define OUT     0x20
#define SEL     0x10
#define NERR    0x08
#define STROBE  0x01

#define INVERT (NBSY | NACK | SEL | NERR)
#define MASK   (NBSY | NACK | OUT | SEL | NERR)
#define NOT_READY(x) ((inb(x)^INVERT)&MASK)

void write_single_character(char c) {
    while (NOT_READY(STATUS)) ;
    outb(DATA, c);
    outb(CONTROL, control | STROBE); /* Assert STROBE */
    outb(CONTROL, control ); /* Clear STROBE */
}
```
The Parallel Port Schematic
Interrupts and Polling

Two ways to get data from a peripheral:

- Polling: “Are we there yet?”
- Interrupts: Ringing Telephone
Interrupts

Basic idea:

1. Peripheral asserts a processor’s interrupt input
2. Processor temporarily transfers control to interrupt service routine
3. ISR gathers data from peripheral and acknowledges interrupt
4. ISR returns control to previously-executing program
Many Different Interrupts

What’s a processor to do?
Many Different Interrupts

What’s a processor to do?
ISR polls all potential interrupt sources, then dispatches handler.
**Intel 8259 PIC**

Prioritizes incoming requests & notifies processor

ISR reads 8-bit interrupt vector number of winner

IBM PC/AT: two 8259s; became standard