Block Devices
A device that stores data in fixed-sized blocks, each uniquely addressed, and can be randomly accessed.

Character Devices
Device that delivers or accepts a stream of characters.

Device Controllers
The electronic component of an I/O unit, in contrast with the physical component.

Memory-Mapped I/O
Separate I/O & memory space  Memory-mapped I/O  Hybrid: both memory-mapped & separate spaces

Dynamic Frequency on XScale

<table>
<thead>
<tr>
<th></th>
<th>CPU Data</th>
<th>CPU Code</th>
<th>CPU Xscale</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
<th>SYST</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
<th>CLC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Setting CPU Freq. in WinCE

```c
// Allocate some space for the virtual reference to CCCR
LPVOID virtCCCR = VirtualAlloc(0, sizeof(DWORD), MEM_RESERVE, PAGE_NOACCESS);
// 0x41300000 is the memory-mapped location of the CCCR register
LPVOID CCCR = (LPVOID)(0x41300000 / 256); // shift by 8 bits for ability to address 2^32 bytes

// Map writing the virtual pointer to the physical address of the CCCR register
VirtualCache((LPVOID)pCCCR, CCCR, closeHandle, MEM_RESERVE | PAGE_NOACCESS | PAGE_WRITECOPY);

// Set the CCCR register with the new speed
*(int *)virtCCCR = new_speed;

// Call the assembly function to actually perform the switch
doSwitch(0x02 | 0x01);
// 0x02 means turbo mode, 0x01 means the clock is being switched

// Clean up memory by freeing the virtual register.
VirtualFree(virtCCCR, 0, MEM_RELEASE);
virtCCCR = NULL;
```

Bus Communication

- CPU
- Memory
- I/O

This port allows I/O devices access into memory

DMA

![DMA Diagram]

Interrupts

- CPU
- Interrupt controller
- Disk
- Keyboard
- Printer
- Bus

1. Device is finished
2. Controller issues interrupt
3. CPU acks interrupt

I/O Software Goals

- Device independence
- Uniform naming
- Error handling
- Synchronous vs. asynchronous transfers
- Buffering
- Sharable vs. dedicated devices

Programmed I/O

User

- Printed page
- Printed page
- Printed page

Kernel User

- Printed page
Polling

copy_from_user (buffer, p, count); // copy into kernel buffer

for (j = 0; j < count; j++) {
    // loop for each char
    *printer_data_reg = p[j]; // output a single character
}

return_to_user();

Interrupt-Driven I/O

System Call

copy_from_user (buffer, p, count);

if (count == 0) {
    unblock_user();
} else {
    *printer_data_reg = p[0];
    count--;
}

scheduler(); // and block user

Interrupt Handler

acknowledge_interrupt();
unblock_user();
return_from_interrupt();

DMA

System Call

copy_from_user (buffer, p, count);
set_up_DMA_controller();
scheduler(); // and block user

Interrupt Handler

acknowledge_interrupt();
unblock_user();
return_from_interrupt();

I/O Software Layers

User-level I/O software & libraries
Device-independent OS software
Device drivers
Interrupt handlers
Hardware

User

Operating system (kernel)

Device Drivers

User space
Kernel space
Rest of the OS
Keyboard driver
Disk driver

Keyboard controller
Disk controller
Driver Interfacing

Non-standard driver interfaces

Standard driver interfaces

Buffering

I/O Request