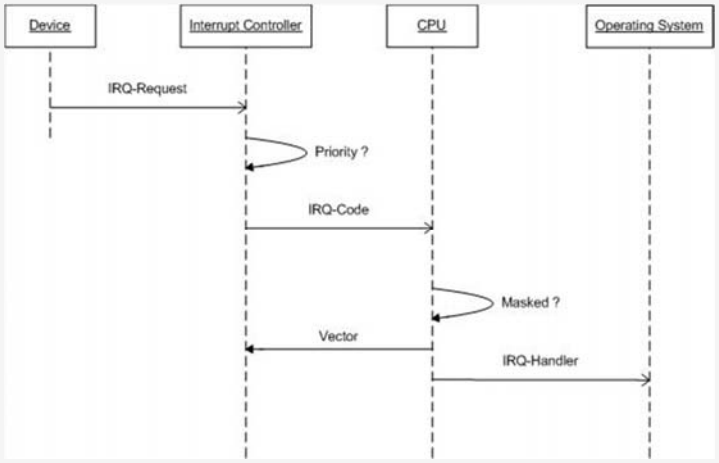
Chapter 13

Interrupt Architecture, Deferred Procedure Calls, Timers and Thread Scheduling

Interrupt Processing Sequence



Interrupt signaling mechanisms

- Line based interrupts
  - Level triggered (e.g. PCI bus)
  - Edge triggered (e.g. legacy ISA bus)
- Message Signaled Interrupt (MSI)
  - Introduced in Vista and above PCI 2.2 spec
  - No sharing / less interrupt latency
  - Enabled by registry subkey of driver's Hardware Key `Interrupt Management \ MessageSignaledInterruptProperties`
  - 2 phase configuration via `IRP_MJ_PNP`
- MSI-X
  - Defined by PCI 3.0 specification
  - Dynamic configuration by direct call interface to bus driver

Interrupt priorities and WDM IRQL

- Hardware abstracted WDM IRQLs
- Only three commonly used interrupt levels
  - `PASSIVE_LEVEL`
  - `APC_LEVEL` (rarely used by 3rd party drivers)
  - `DISPATCH_LEVEL`
  - `DIRQL`
- Explicit raising and lowering
  - `KeRaiseIrql(...)` and `KeLowerIrql(...)`
  - Never drop lower than initial IRQL
  - Not usable for synchronization
- DON'T on IRQL >= `DISPATCH_LEVEL`
  - DON'T touch paged memory
  - DON't wait for dispatcher objects

Interrupt processing

- Interrupt service routine (ISR)
  - Check – is interrupt coming from own device?
  - Communicate with device to stop it from interrupting
- `DIRQL` interrupt processing
  - Minimize CPU cycles spent on `DIRQL`
  - Queue DPC (Deferred Procedure Call) for interrupt post processing at `DISPATCH_LEVEL`
- `PASSIVE_LEVEL` interrupt processing
  - GPIO secondary interrupts
  - ISR can access devices on SPB at `PASSIVE_LEVEL`
  - ISR can wait

DPCs and DPC Queues

- DPC Queues (one per processor)
  - DPC object only queued once (per processor)
  - DPCs can run in parallel in multiprocessor environment
- Queue insertion
  - On same processor (by default)
  - Specific processor by: `KeSetTargetProcessorDpc(...)`
  - End of the queue (by default)
  - Beginning of the queue by: `KeSetImportanceDpc(...)`

Threaded DPCs

- Supported by Vista and above
- Activation / DeactivationThreaded DPCs
  - Enabled by default
  - `HKLM\System\CCS\Control\SessionManager\Kernel\ThreadDpcEnable`
  - `KeInitializeThreadedDpc(...)`
- Programming
  - Executed at `IRQL PASSIVE_LEVEL`
  - Code must still obey same rules as if running on IRQL `DISPATCH_LEVEL`

Timers

- Timers DPCs
  - `KeInitializeTimer(...)`
    `KeSetTimer(...)`
  - 10 milliseconds granularity
- Coalescable timers
  - Available on Windows 7 and above
  - Power saving
  - `KeSetCoalescableTimer(...)`

Windows Thread Scheduling

Thread execution context

- What makes kernel code run?
- Context of user mode application
  - Request handling only at top level of device stack WDF Driver Object callback `EvtIoInCallerContext`
  - Driver created system thread in application context `PsCreateSystemThread(…)`
- System worker threads
  - System thread pool calling e.g. calling `DriverEntry`
  - Driver created system thread in system context `PsCreateSystemThread(…)`
- Arbitrary context on elevated IRQL