Exceptions

- Normal program flow: beginning to end
  - can only evaluate pre-programmed conditional decisions
  - cannot respond to exceptions in normal program flow

  Example:

- Exceptions = break in normal program flow
  2 Types
  - Resets:
    Example:

  - Interrupts:

    Example:
 Resets

- **Purpose**: return to known state of operation
- **Cause**: external event or malfunction
- **Events that trigger system reset (HC12)**
  - **Power-on Reset**
    - **Cause**:
      - Action: start program execution with defined configuration
  - **Computer Operating Properly (COP) Reset**
    - COP is a hardware unit that detects malfunctions in software execution
      - user programmable “time out” duration
    - COP must be enabled; generally disabled while testing
    - **Example**:
      - **Clock Monitor Reset**
        - **Cause**:
      - **External Reset**
        - **Cause**:
          - e.g. push button reset, as used in lab
Reset Response

When a reset occurs…

▪ For external resets
  ▪ program starts over from beginning

▪ For internal resets (COP & Clock Monitor)
  ▪ specific “reset vector” immediately stored to the PC
  ▪ jump to user code (@ reset vector)
  ▪ reset vector
    ▪ a preset memory address assigned to each exception event
    ▪ contains address of user-defined code to manage exception event
  ▪ See “exception vector” tables at end of slides

Interrupts

▪ Purpose
  ▪ permit MCU to respond to events of higher priority than the normal program

▪ Cause
  ▪ signal or event causing a break in normal program flow

▪ Action
  ▪ jump to “interrupt vector” (like reset vector)
  ▪ “interrupt vector” points to specific subroutine for each interrupt event

▪ Alternative to Interrupts
  ▪ Polling: code continuously monitors for event (e.g., hardware flag) to occur; can’t do anything else except wait for event
Types of Interrupts

- **Maskable**
  - interrupt can be enabled/disabled by software programmable hardware settings
    - e.g., configuration registers (register set)
- **Non-Maskable**
  - interrupt request always honored; can not be turned off by user
  - useful for critical events
    - e.g., power down for batter level low

**HC12/S12 Interrupt System**

**Non-Maskable Interrupts**

- **Indicator/Control**
  - active when CCR bit X is ‘0’
  - not controllable by user
- **Events**
  - Unimplemented Instruction Trap
  - for unassigned op-codes
  - Software Interrupt Instruction (SWI)
    - ASM instruction
  - Nonmaskable Interrupt Request (XIRQ)
    - active low external pin on MCU
HC12/S12 Interrupt System

- Maskable Interrupts
- Control/Indicator
  - master “mask” (on/off switch): CCR bit I (I = inhibit)
  - @ system reset, I = 1 (off)
  - CLI instruction → I = 0 (turns maskable interrupt system on)
    - CLI =
  - SEI instruction → I = 1 (turns maskable interrupt system off)
    - SEI =
- Local Masks
  - MCU hardware system interrupts are enabled by local masks
    - e.g., TOI bit in the timer system enables timer interrupts

---

HC12/S12 Maskable Interrupts

- Maskable Interrupt Request (IRQ)
  - external pin, active low
  - primary external interrupt
    - can be configured for multiple interrupt sources
- Real-Time Interrupt (RTI)
  - periodic interrupt, programmable duration
  - Example:
- Timer Overflow
  - enabled by TOI bit (in configuration register)
- Analog to Digital System
  - create interrupt when A/D conversion complete
- Serial Communication System
  - several maskable interrupts

- Many more…
  
  see complete list in exception vector list (end of slides)
Interrupt Response

What happens when an interrupt occurs?

- **Interrupt Service Routine (ISR)**
  - software code to handle each interrupt source
  - address (location) for each ISR stored at specific memory location called an interrupt vector
  - review exception vector list (end of slides)

- **RTI (return from interrupt)**
  - ASM instruction to return from interrupt back to main program

- **Automated Interrupt Functions**
  - performed automatically by hardware (without software)
  - varies with MCU
  - see flowchart on next slide

---

Interrupt Response Chart

**HC12 Auto Interrupt Response**

1. Finish current instruction
   - PC set to next instruction

2. Prepare for ISR
   - calculate return address
   - push return address to stack
   - push register values to stack
   - disable interrupts

3. Fetch ISR vector
   - load PC with ISR vector
   - execute ISR
   - return to main (RTI) within ISR

- see stack order on next slide
Exception Priority

What happens if multiple exceptions occur at the same time?

- Exceptions have hard-wired priority (order of importance) that determines order

1. Resets
   - Power-on reset
   - Clock monitor reset
   - COP watchdog reset

2. Non-maskable interrupts (NMI)
   - Unimplemented instruction trap
   - SWI
   - XIRQ

3. Maskable interrupts
   - Priority set by interrupt vector map

   higher addresses = higher priority

Hardware Interfacing

- Example: backup power system; interrupt driven switching between two rechargeable power supplies
  - IRQ'
  - PortG, pin 5,
  - PortG, pin 0,
### 68HCS12 Exception Vector Locations

<table>
<thead>
<tr>
<th>Vector Address</th>
<th>Interrupt Source</th>
<th>CCR Mask</th>
<th>Local Enable</th>
<th>HPRIO Value to Elevate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5FFFE, 5FFFF</td>
<td>Reset</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>5FFEC, 5FFFFD</td>
<td>Clock Monitor</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>5FFFA, 5FFFB</td>
<td>CCP failure</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>5FFED, 5FFFE</td>
<td>Unexpected</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>S'FFFC, S'FFFFD</td>
<td>RWI</td>
<td>X-Bit</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>S'FFFA, S'FFFB</td>
<td>IRQ</td>
<td>I-Bit</td>
<td>IRQCR (IRQEN)</td>
<td>$F2</td>
</tr>
<tr>
<td>S'FFED, S'FFFE</td>
<td>Real Time Intertu</td>
<td>I-Bit</td>
<td>CRGINT (RTIE)</td>
<td>$F0</td>
</tr>
<tr>
<td>S'FFED, S'FFFE</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C0)</td>
<td>$EE</td>
</tr>
<tr>
<td>S'FFEC, S'FFFD</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C1)</td>
<td>$EC</td>
</tr>
<tr>
<td>S'FFDA, S'FFDB</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C2)</td>
<td>$EA</td>
</tr>
<tr>
<td>S'FFEB, S'FFEC</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C3)</td>
<td>$ED</td>
</tr>
<tr>
<td>S'FFEB, S'FFED</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C4)</td>
<td>$EB</td>
</tr>
<tr>
<td>S'FFEB, S'FFEE</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C5)</td>
<td>$E4</td>
</tr>
<tr>
<td>S'FFED, S'FFEF</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C6)</td>
<td>$E2</td>
</tr>
<tr>
<td>S'FFED, S'FFFG</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TIE (C7)</td>
<td>$E0</td>
</tr>
<tr>
<td>S'FFEC, S'FFFD</td>
<td>Enhanced Capture</td>
<td>I-Bit</td>
<td>TERG2 (TOF)</td>
<td>$DE</td>
</tr>
<tr>
<td>S'FFEC, S'FFFD</td>
<td>Pulse accumulat</td>
<td>I-Bit</td>
<td>PACTL (PAOV1)</td>
<td>$DC</td>
</tr>
<tr>
<td>S'FFEC, S'FFFD</td>
<td>Pulse accumulat</td>
<td>I-Bit</td>
<td>PACTL (PAV1)</td>
<td>$DA</td>
</tr>
<tr>
<td>S'FFEA, S'FFEB</td>
<td>SM2</td>
<td>I-Bit</td>
<td>SPCH1 (SPH, SPTE)</td>
<td>$UX</td>
</tr>
<tr>
<td>S'FFDA, S'FFDB</td>
<td>SC2</td>
<td>I-Bit</td>
<td>SCOCR2 (TIE, TCE, RIE, ILE)</td>
<td>$DB</td>
</tr>
<tr>
<td>S'FFDC, S'FFDD</td>
<td>SC1</td>
<td>I-Bit</td>
<td>SCOCR2 (TIE, TCE, RIE, ILE)</td>
<td>$DA</td>
</tr>
<tr>
<td>S'FFDC, S'FFDD</td>
<td>ATDOCT2 (ASCIE)</td>
<td>I-Bit</td>
<td>ATDOCT2 (ASCIE)</td>
<td>$DO</td>
</tr>
<tr>
<td>S'FFDA, S'FFDB</td>
<td>Port J</td>
<td>I-Bit</td>
<td>PTIUP (PTIF)</td>
<td>$CC</td>
</tr>
<tr>
<td>S'FFDA, S'FFDB</td>
<td>Port M</td>
<td>I-Bit</td>
<td>PTIUP (PTI)</td>
<td>$CC</td>
</tr>
<tr>
<td>S'FFDA, S'FFDB</td>
<td>Modem Down Counter</td>
<td>I-Bit</td>
<td>NCU/NULL (NZX)</td>
<td>$CA</td>
</tr>
</tbody>
</table>

---

**Note:** The above table lists the exception vector locations for the 68HCS12 microcontroller. Each entry specifies the vector address, interrupt source, CCR mask, local enable, and HPRIO value to elevate.