Interrupts

- An interrupt can be compared with a phone call interrupting your task which you will resume when the call is finished.
- You can mask an interrupt just as you can decide not to answer any phone calls.
- Especially by turning off your phones.
- You will not return from certain interrupts (e.g., reset).
- Un-maskable interrupts have highest priority and can be compared to fire alarm.
- You usually do not go back to continue your work after a fire has been detected.

Example of Interrupts

- Hardware Reset (RESET)
- External Device (IRQ, XIRQ)
- Asynchronous Serial Interface (SCI)
- Synchronous Serial Interface (SPI)
- Timer Input Capture (TIC)
- Timer Output Compare (TOC)
- Pulse Accumulator
- Real-time Interrupt, Watchdog Timer

HCS12 Interrupts

- 17 hardware interrupts
- 1 software interrupt (used for debugging)

Maskable vs. Nonmaskable Interrupts

- Maskable Interrupts (122)
  - Enabled or disabled by software
  - The I bit in the CCR turns them on or off
- Non-maskable Interrupts (6)
  - XIRQ
  - Illegal Opcode Trap
  - Software interrupt (SWI instruction), Background debugging
  - Reset, CMR, COP
**Basic Interrupt Behavior**

- Current processor context will be stored when interrupt occurs
  - Push PC, Y, X, B:A (D), and CCR to stack

**HCS12 Interrupts – IRQ, XIRQ**

- IRQ & XIRQ are external pins
- Can connect several devices to interrupt pins
  - If you hook up more than one, you will need to check which one caused the interrupt
  - I interrupt -> I bit set
  - X interrupt -> X bit set
  - I or X interrupt can bring processor out of Wait/Sleep mode

**CCR - Condition Code Register**

- SXHINZVC
  - X interrupt
  - I interrupt

RTI is always the last command of interrupt routine:
Will restore all registers previously saved on the stack
HCS12 Non-maskable Interrupts
- Reset or POR
- Clock Monitor Reset
- COP Watchdog
- Non-maskable Interrupt XIRQ
- Unimplemented/Illegal Opcode
- Software interrupt instruction (used by background debugger)

HCS12 - XIRQ
- XIRQ is a pseudo non-maskable interrupt
- X bit in CCR is initially set to 1 on reset
  - Inhibits all interrupts on XIRQ pin
- X bit be set to 0 from software
  - XIRQ now acts as a non-maskable interrupt
  - X can no longer set to 1

HCS12 - Maskable Interrupts
- The following interrupts are maskable using the I bit of the CCR register
  - I bit in CCR is initially set to 1 on reset
    - All makable interrupts are ignored
    - Can enable/disable maskable interrupts by writing a 0/1 to I
  - clr - Clears I bit
  - set - Set I bit
- SCI, receive data register full, receive overrun, idle line, transmit register empty, transmit complete
- SPI, transfer complete
- Timer circuit, accumulator, timer overflow, output compare, input capture

HCS12 Interrupt Vector Map
- Defines the program location at which the interrupt service code is located for each interrupt
- Example:
  - Interrupt vector for the system reset vector is the usually starting location of ROM
    - $FFFE-$FFFF = $4000

<table>
<thead>
<tr>
<th>Vector Address</th>
<th>Interrupt Source</th>
<th>CCR Mask</th>
<th>Local Enable</th>
<th>HIPIO Value to Decode</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FFFE, $FFFF</td>
<td>External Reset, Power On Reset or Low Voltage RESET (see COP Flags Register to determine reset source)</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
<tr>
<td>$FFE0, $FFE1</td>
<td>Clock Monitor fail reset</td>
<td>None</td>
<td>COP1 TP (COP1P)</td>
<td>–</td>
</tr>
<tr>
<td>$FFE5, $FFE6</td>
<td>COP failure reset</td>
<td>None</td>
<td>COP fault select</td>
<td>–</td>
</tr>
<tr>
<td>$FFE2, $FFE3</td>
<td>Unimplemented instruction trap</td>
<td>None</td>
<td>None</td>
<td>–</td>
</tr>
</tbody>
</table>
### HCS12 Interrupt Vector Map

<table>
<thead>
<tr>
<th>Interrupts</th>
<th>Name</th>
<th>Priority</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0x01</td>
<td>BUR</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>0x02-0x03</td>
<td>XIRQ</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>0x04-0x07</td>
<td>RQ</td>
<td>3-4</td>
<td>None</td>
</tr>
<tr>
<td>0x08-0x0F</td>
<td>GIRQ1</td>
<td>5-6</td>
<td>None</td>
</tr>
<tr>
<td>0x10-0x17</td>
<td>GIRQ2</td>
<td>7-8</td>
<td>None</td>
</tr>
<tr>
<td>0x18-0x1F</td>
<td>GIRQ3</td>
<td>9-10</td>
<td>None</td>
</tr>
<tr>
<td>0x20-0x27</td>
<td>GIRQ4</td>
<td>11-12</td>
<td>None</td>
</tr>
<tr>
<td>0x28-0x2F</td>
<td>GIRQ5</td>
<td>13-14</td>
<td>None</td>
</tr>
<tr>
<td>0x30-0x37</td>
<td>SPI</td>
<td>15-16</td>
<td>None</td>
</tr>
<tr>
<td>0x38-0x3F</td>
<td>SCI</td>
<td>17-18</td>
<td>None</td>
</tr>
</tbody>
</table>
Interrupts Continued

What is the Interrupt Vector Table?

Let’s assume we want to send or receive data with interrupts. Our main program does not know when data will arrive from external device.
How do we pass data from the interrupt to the main program?

Main Program
Interrupt Service routine
Producer/consumer problem with an input device

How do we pass data from the main program to the interrupt?

FIFO

Main Purpose of Interrupts: Dealing with Asynchronous Events
Example Interrupt

• Wildcat Song Example
  – What programming elements can you find?