Interrupt-Driven Input/Output

Textbook: Chapter 3, Section 3.1 (Cortex registers & op. modes)
Chapter 9, Section 9.2 (Interrupt concepts)
ARM Cortex-M4 User Guide (Interrupts, exceptions, NVIC)
Outline

- Interrupt vectors and vector table
- Interrupt masks and priorities
- Cortex Nested Vectored Interrupt Controller (NVIC)
- STM32F4 external interrupt signals (EXTI0 – EXTI15)
- System design when interrupts used
Prioritized, vectored interrupts

Priorities determine what interrupt gets the CPU first.

Vectors determine what code is called for each type of interrupt.
Interrupt vectors

- **Interrupt vector** = address of handler function
  - Allow different devices to be handled by different code.
- **Interrupt vector table:**
  - Directly supported by CPU architecture and/or
  - Supported by a separate interrupt-support device/function

<table>
<thead>
<tr>
<th>Interrupt vector table head</th>
<th>address of handler 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>address of handler 1</td>
</tr>
<tr>
<td></td>
<td>address of handler 2</td>
</tr>
<tr>
<td></td>
<td>address of handler 3</td>
</tr>
</tbody>
</table>
## Cortex processor exceptions

<table>
<thead>
<tr>
<th>Priority</th>
<th>IRQ#</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>-3</td>
<td>Power-up or warm reset</td>
</tr>
<tr>
<td>NMI</td>
<td>-2</td>
<td>-14 Non-maskable interrupt from peripheral or software</td>
</tr>
<tr>
<td>HardFault</td>
<td>-1</td>
<td>-13 Error during exception processing or no other handler</td>
</tr>
<tr>
<td>MemManage</td>
<td>Config</td>
<td>-12 Memory protection fault (MPU-detected)</td>
</tr>
<tr>
<td>BusFault</td>
<td>Config</td>
<td>-11 AHB data/prefetch aborts</td>
</tr>
<tr>
<td>UsageFault</td>
<td>Config</td>
<td>-10 Instruction execution fault - undefined instruction, illegal unaligned access</td>
</tr>
<tr>
<td>SVCcall</td>
<td>Config</td>
<td>-5 System service call (SVC) instruction</td>
</tr>
<tr>
<td>DebugMonitor</td>
<td>Config</td>
<td>Break points/watch points/etc.</td>
</tr>
<tr>
<td>PendSV</td>
<td>Config</td>
<td>-2 Interrupt-driven request for system service</td>
</tr>
<tr>
<td>SysTick</td>
<td>Config</td>
<td>-1 System tick timer reaches 0</td>
</tr>
<tr>
<td>IRQ0</td>
<td>Config</td>
<td>0 Signaled by peripheral or by software request</td>
</tr>
<tr>
<td>IRQ1 (etc.)</td>
<td>Config</td>
<td>1 Signaled by peripheral or by software request</td>
</tr>
</tbody>
</table>

Lowest priority number = highest priority

IRQ #s are used in CMSIS function calls

Vendor peripheral Interrupts
IRQ0 .. IRQ239
Cortex interrupt vector table

- 32-bit vector (handler address) loaded from table into PC (while saving CPU context)
- Peripherals use positive IRQ #s (up to 240 allowed by Cortex - STM32F407 uses 82)
- CPU exceptions use negative IRQ #s
- IRQ # used in CMSIS function calls
- Exception # stored in CPU PSR
- Reset vector includes initial stack pointer

<table>
<thead>
<tr>
<th>Exception number</th>
<th>IRQ number</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>16+n</td>
<td>n</td>
<td>0x0040+4n</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0x004C</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>0x0048</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>0x0044</td>
</tr>
<tr>
<td>15</td>
<td>-1</td>
<td>0x0040</td>
</tr>
<tr>
<td>14</td>
<td>-2</td>
<td>0x003C</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>0x0038</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>-5</td>
<td>0x002C</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-11</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-13</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-14</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- IRQn:
  - IRQ0
  - IRQ1
  - IRQ2
  - Systick
  - PendSV
  - Reserved
  - Reserved for Debug
  - SVC
  - Usage fault
  - Bus fault
  - Memory management fault
  - Hard fault
  - NMI
  - Reset
  - Initial SP value
<table>
<thead>
<tr>
<th>Position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>-3</td>
<td>fixed</td>
<td>Reset</td>
<td>Reset</td>
<td></td>
<td>0x0000 0004</td>
</tr>
<tr>
<td>6</td>
<td>settable</td>
<td>SysTick</td>
<td>System tick timer</td>
<td></td>
<td>0x0000 003C</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>settable</td>
<td>WWDG</td>
<td>Window Watchdog interrupt</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>settable</td>
<td>PVD</td>
<td>PVD through EXTI line detection interrupt</td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>settable</td>
<td>TAM_P_TAMP</td>
<td>Tamper and TimeStamp interrupts through the EXTI line</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>settable</td>
<td>RTC_WKUP</td>
<td>RTC Wakeup interrupt through the EXTI line</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>settable</td>
<td>FLASH</td>
<td>Flash global interrupt</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>settable</td>
<td>RCC</td>
<td>RCC global interrupt</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>settable</td>
<td>EXTI0</td>
<td>EXTI Line0 interrupt</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>settable</td>
<td>EXTI1</td>
<td>EXTI Line1 interrupt</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>settable</td>
<td>EXTI2</td>
<td>EXTI Line2 interrupt</td>
<td>0x0000 0060</td>
</tr>
<tr>
<td>9</td>
<td>16</td>
<td>settable</td>
<td>EXTI3</td>
<td>EXTI Line3 interrupt</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>settable</td>
<td>EXTI4</td>
<td>EXTI Line4 interrupt</td>
<td>0x0000 0068</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>settable</td>
<td>DMA1_Stream0</td>
<td>DMA1 Stream0 global interrupt</td>
<td>0x0000 006C</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>settable</td>
<td>DMA1_Stream1</td>
<td>DMA1 Stream1 global interrupt</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>settable</td>
<td>DMA1_Stream2</td>
<td>DMA1 Stream2 global interrupt</td>
<td>0x0000 0074</td>
</tr>
</tbody>
</table>
STM32F4 vector table from startup code (partial)

<table>
<thead>
<tr>
<th>Address</th>
<th>Vector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__Vectors</td>
<td>DCD __initial_sp</td>
<td>Top of Stack</td>
</tr>
<tr>
<td>DCD</td>
<td>Reset_Handler</td>
<td>Reset Handler</td>
</tr>
<tr>
<td>DCD</td>
<td>NMI_Handler</td>
<td>NMI Handler</td>
</tr>
<tr>
<td>DCD</td>
<td>SVC_Handler</td>
<td>SVCall Handler</td>
</tr>
<tr>
<td>DCD</td>
<td>DebugMon_Handler</td>
<td>Debug Monitor Handler</td>
</tr>
<tr>
<td>DCD</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>DCD</td>
<td>PendSV_Handler</td>
<td>PendSV Handler</td>
</tr>
<tr>
<td>DCD</td>
<td>SysTick_Handler</td>
<td>SysTick Handler</td>
</tr>
</tbody>
</table>

; External Interrupts
<table>
<thead>
<tr>
<th>Address</th>
<th>Vector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCD</td>
<td>WWDG_IRQHandler</td>
<td>Window WatchDog</td>
</tr>
<tr>
<td>DCD</td>
<td>PVD_IRQHandler</td>
<td>PVD via EXTI Line detection</td>
</tr>
<tr>
<td>DCD</td>
<td>TAMPS_TAMP_IRQHandler</td>
<td>Tamper/TimeStamps via EXTI</td>
</tr>
<tr>
<td>DCD</td>
<td>RTC_WKUP_IRQHandler</td>
<td>RTC Wakeup via EXTI line</td>
</tr>
<tr>
<td>DCD</td>
<td>FLASH_IRQHandler</td>
<td>FLASH</td>
</tr>
<tr>
<td>DCD</td>
<td>RCC_IRQHandler</td>
<td>RCC</td>
</tr>
<tr>
<td>DCD</td>
<td>EXTI0_IRQHandler</td>
<td>EXTI Line0</td>
</tr>
<tr>
<td>DCD</td>
<td>EXTI1_IRQHandler</td>
<td>EXTI Line1</td>
</tr>
<tr>
<td>DCD</td>
<td>EXTI2_IRQHandler</td>
<td>EXTI Line2</td>
</tr>
</tbody>
</table>
Prioritized interrupts

- Up to 256 priority levels
- 8-bit priority value
- Implementations may use fewer bits
  STM32F4xx uses upper 4 bits of each priority byte => 16 levels
- NMI & HardFault priorities are fixed
- Lowest # = Highest priority
Program Status Register

Access 3-parts separately or all at once

Figure 3. APSR, IPSR and EPSR bit assignments

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23</th>
<th>16 15</th>
<th>10 9 8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSR</td>
<td>N Z C V Q</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>IPSR</td>
<td>Reserved</td>
<td>ISR_NUMBER</td>
<td></td>
</tr>
<tr>
<td>EPSR</td>
<td>Reserved</td>
<td>ICI/IT</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Figure 4. PSR bit assignments

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23</th>
<th>16 15</th>
<th>10 9 8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>N Z C V Q ICI/IT T</td>
<td>Reserved</td>
<td>ICI/IT</td>
<td>ISR_NUMBER</td>
</tr>
</tbody>
</table>

ISR_NUMBER = # of current exception

Q = Saturation, T = Thumb bit

ARM instructions to “access special registers”

MRS Rd,spec ;move from special register (other than R0-R15) to Rd
MSR spec,Rs ;move from register Rs to special register
## Interrupt Program Status Register (ISPR)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:9</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| Bits 8:0 | ISR_NUMBER:
This is the number of the current exception:  
0: Thread mode
1: Reserved
2: NMI
3: Hard fault
4: Memory management fault
5: Bus fault
6: Usage fault
7: Reserved
....  
10: Reserved
11: SVCcall
12: Reserved for Debug
13: Reserved
14: PendSV
15: SysTick
16: IRQ0\(^{(1)}\)
.... |  

- **No active interrupt**
- **Cortex CPU interrupts**
- **User (vendor) interrupts IRQ0 – IRQ239**
CPU Priority Mask Register

**Figure 5.** PRIMASK bit assignments

PRIMASK = 1 prevents (masks) activation of all exceptions with configurable priority
PRIMASK = 0 permits (enables) exceptions

**Special Cortex-M Assembly Language Instructions**

- CPSIE I ; Change Processor State/Enable Interrupts (sets PRIMASK = 0)
- CPSID I ; Change Processor State/Disable Interrupts (sets PRIMASK = 1)

Execute CPSIE I in a program to enable interrupts, after other initialization done.
ARM Cortex-M Interrupts

In the Device:

- Each potential interrupt source has a separate arm (enable) bit
  - Set for those devices from which interrupts, are to be accepted
  - Deactivate in those devices from which interrupts are not allowed
- Each potential interrupt source has a separate flag bit
  - hardware sets the flag when it wishes to request an interrupt (an “event” occurs)
  - software clears the flag in ISR to signify it is processing the request
  - flags can be tested by software if interrupts not desired

In the CPU:

- Cortex-M CPUs receive interrupt requests via the Nested Vectored Interrupt Controller (NVIC)
  - NVIC sends highest priority request to the CPU
- Interrupt enable conditions in processor
  - Global interrupt enable bit, I, in PRIMASK register
  - Priority level, BASEPRI, of allowed interrupts (0 = all)
Interrupt Conditions

- Four conditions must be true simultaneously for an interrupt to occur:
  1. **Arm**: control bit for each possible source is set within the peripheral device
  2. **Enable**: interrupts globally enabled in CPU (I=0 in PRIMASK)
  3. **Level**: interrupt level must be less than BASEPRI
  4. **Trigger**: hardware action sets source-specific flag in the peripheral device

- **Interrupt remains pending** if trigger is set but any other condition is not true
  - Interrupt serviced once all conditions become true

- Need to **acknowledge** interrupt
  - Clear trigger flag or will get endless interrupts!
Nested Vectored Interrupt Controller (NVIC)

- Hardware unit that coordinates interrupts from multiple sources
  - Separate enable flag for each interrupt source (NVIC_ISERx)
  - Define priority level of each interrupt source (NVIC_IPRx)
  - Set/clear interrupts to/from pending state
  - Trigger interrupts through software

- Higher priority interrupts can interrupt lower priority ones
  - Lower priority interrupts are not sent to the CPU until higher priority interrupt service has been completed
NVIC Interrupt Enable Registers

- Three “set interrupt enable” registers –
  \texttt{NVIC\_ISER0}, \texttt{NVIC\_ISER1}, \texttt{NVIC\_ISER2}
  - Each 32-bit register has a single enable bit for a particular device
  - Write 1 to a bit to set the corresponding interrupt enable bit
  - Writing 0 has no effect

<table>
<thead>
<tr>
<th>Register</th>
<th>IRQ numbers</th>
<th>Interrupt numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIC_ISER0</td>
<td>0-31</td>
<td>16-47</td>
</tr>
<tr>
<td>NVIC_ISER1</td>
<td>32-63</td>
<td>48-79</td>
</tr>
<tr>
<td>NVIC_ISER2</td>
<td>64-81</td>
<td>80-97</td>
</tr>
</tbody>
</table>

- Three corresponding “clear interrupt enable” registers
  \texttt{NVIC\_ICER0}, \texttt{NVIC\_ICER1}, \texttt{NVIC\_ICER2}
  - Write 1 to clear the interrupt enable bit (disable the interrupt)
  - Writing 0 has no effect
NVIC interrupt priority registers

NVIC_IPRx (x=0..20) – Interrupt Priority Registers

- 8-bit priority field for each interrupts (4-bit field in STM32F4)
  - Four 8-bit priority values per register
  - 0 = highest priority level
  - IPR Register# x = IRQ# DIV 4
  - Byte offset within the IPR register = IRQ# MOD 4

Example: IRQ45

- 45/4 = 11 with remainder 1 (register NVIC_IPR11, byte offset 1)
  - Write priority<<8 to NVIC_IPR11
- 45/32 = 1 with remainder 13:
  - Write 1<<13 to NVIC_ISER1
NVIC register addresses

NVIC_ISER0/1/2 = 0xE000E100/104/108
NVIC_ICER0/1/2 = 0xE000E180/184/188
NVIC_IPR0/1/2/…/20 = 0xE00E400/404/408/40C/….500

;Example – Enable EXTI0 with priority 5 (EXTI0 = IRQ6)
NVIC_ISER0  EQU  0xE000E100          ;bit 6 enables EXTI0
NVIC_IPR1    EQU  0xE000E404  ;3rd byte = EXTI0 priority

  ldr  r0,=NVIC_ISER0
  mov  r1,#0x0040 ;Set bit 6 of ISER0 for EXTI0
  str  r1,[r0]
  ldr  r0,=NVIC_IPR1  ;IRQ6 priority in IPR1[23:16]
  ldr  r1,[r0]
  bic  r1,#0x00ff0000 ;Clear [23:16] for IRQ6
  orr  r1,#0x00500000 ;Bits [23:20] = 5
  str  r1,[r0] ;Upper 4 bits of byte = priority
CMSIS\(^1\) functions

- NVIC_Enable\(\text{IRQn\_Type IRQn}\)
- NVIC_Disable\(\text{IRQn\_Type IRQn}\)
- NVIC_SetPending\(\text{IRQn\_Type IRQn}\)
- NVIC_ClearPending\(\text{IRQn\_Type IRQn}\)
- NVIC_GetPending\(\text{IRQn\_Type IRQn}\)
- NVIC_SetPriority\(\text{IRQn\_Type IRQn,unit32\_t priority}\)
- NVIC_GetPriority\(\text{IRQn\_Type IRQn}\)

\(^1\)CMSIS = Cortex Microcontroller Software Interface Standard

- Vendor-independent hardware abstraction layer for Cortex-M
- Facilitates software reuse
- Other CMSIS functions: System tick timer, Debug interface, etc.
STM32F4 external interrupt/event controller

23 edge detectors to trigger events and interrupts signaled by 240 GPIO pins and 7 internal events.
STM32F4 external interrupt sources

- 16 multiplexers: one each for EXTI0..EXTI15.
- Mux x select bit x of Port “N” as EXTIx.
  \[ x = 0..15 \]
  \[ “N” = 0..8 \text{ for ports A..I} \]

SYSCFG_EXTICR1

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>12</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EXTI3</td>
<td>EXTI2</td>
<td>EXTI1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td></td>
<td>EXTI0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STM32F4 EXTI Registers

- 23 bits in each register - control 23 interrupts/events

- EXTI_IMR – interrupt mask register
  - 0 masks (disables) the interrupt
  - 1 unmasks (enables) the interrupt

- EXTI_RTSR/FTSR – rising/falling trigger selection register
  - 1 to enable rising/falling edge to trigger the interrupt/event
  - 0 to ignore the rising/falling edge

- EXTI_PR – interrupt/event pending register
  - read 1 if interrupt/event occurred
  - clear bit by writing 1 (writing 0 has no effect)
  - write 1 to this bit in the interrupt handler to clear the pending state of the interrupt
Example: Enable EXTI0 as rising-edge triggered

;System Configuration Registers
SYSCFG   EQU      0x40013800
EXTICR1  EQU      0x08
;External Interrupt Registers
EXTI     EQU      0x40013C00
IMR      EQU      0x00  ;Interrupt Mask Register
RTSR     EQU      0x08  ;Rising Trigger Select
FTSR     EQU      0x0C  ;Falling Trigger Select
PR       EQU      0x14  ;Pending Register

;select PC0 as EXTI0
ldr     r1,=SYSCFG ;SYSCFG selects EXTI sources
ldrh    r2,[r1,#EXTICR1] ;EXTICR1 = sources for EXTI0 - EXTI3
bic     r2,#0x000f ;Clear EXTICR1[3-0] for EXTI0 source
orr     r2,#0x0002 ;EXTICR1[3-0] = 2 to select PC0 as EXTI0 source
strh    r2,[r1,#EXTICR1] ;Write to select PC0 as EXTI0

;configure EXTI0 as rising-edge triggered
ldr     r1,=EXTI ;EXTI register block
mov     r2,#1 ;bit #0 for EXTI0 in each of the following registers
str     r2,[r1,#RTSR] ;Select rising-edge trigger for EXTI0
str     r2,[r1,#PR] ;Clear any pending event on EXTI0
str     r2,[r1,#IMR] ;Enable EXTI0
Interrupt Rituals

• Things you must do in every ritual
  • Initialize data structures (counters, pointers)
  • Arm interrupt in the peripheral device
    • Enable a flag to trigger an interrupt
    • Clear the flag (to ignore previous events)
  • Configure NVIC
    • Enable interrupt (NVIC_ISERx)
    • Set priority (NVIC_IPRx)
  • Enable CPU Interrupts
    • Assembly code      CPSIE   I
    • C code            EnableInterrupts();
Interrupt Service Routine (ISR)

- Things you must do in every interrupt service routine
  - Acknowledge
    - clear flag that requested the interrupt
    - SysTick is exception; automatic acknowledge
  - Maintain contents of R4-R11 (AAPCS)
  - Communicate via shared global variables
Sources of interrupt overhead

- Handler execution time.
- Interrupt mechanism overhead.
- Register save/restore.
- Pipeline-related penalties.
- Cache-related penalties.
- **Interrupt “latency”** = time from activation of interrupt signal until event serviced.
- ARM worst-case latency to respond to interrupt is 27 cycles:
  - 2 cycles to synchronize external request.
  - Up to 20 cycles to complete current instruction.
  - 3 cycles for data abort.
  - 2 cycles to enter interrupt handling state.