CP316
Interrupts

Terry Sturtevant

Wilfrid Laurier University

June 27, 2019

Introduction to Interrupts

Introduction to Interrupts

Interrupts allow program control to change due to *events*.

Introduction to Interrupts

Interrupts allow program control to change due to *events*.

Interrupt vectors are program locations

Introduction to Interrupts

Interrupts allow program control to change due to *events*.

Interrupt vectors are program locations

where **interrupt service routines** are located.

Introduction to Interrupts

Interrupts allow program control to change due to *events*.

Interrupt vectors are program locations

where **interrupt service routines** are located.

After interrupt service, the program control returns to original.
but not the other way around.

Terminology

Terminology

- **Enabled/Disabled**

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred
- **Flag**

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred
- **Flag**

  bit in memory indicating an interrupt is pending

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred
- **Flag**

  bit in memory indicating an interrupt is pending

  Note: Flags are usually set whether or not corresponding interrupts are enabled.

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred
- **Flag**

  bit in memory indicating an interrupt is pending

  Note: Flags are usually set whether or not corresponding interrupts are enabled.
- **Set** or **Clear**

Terminology

- **Enabled/Disabled**

  whether a specific event will generate an interrupt
- **Pending**

  whether the event has occurred
- **Flag**

  bit in memory indicating an interrupt is pending

  Note: Flags are usually set whether or not corresponding interrupts are enabled.
- **Set** or **Clear**

  state of flag indicating pending or not

Interrupts

Interrupts

overview

Interrupts

overview

→ Chapters 12 and 13

Here is a compiled code fragment from a typical program:

Here is a compiled code fragment from a typical program:

```
;;; vectors
    rjmp RESET ; Reset Handler
    rjmp EXT_INT0 ; IRQ0 Handler
    rjmp EXT_INT1 ; IRQ1 Handler
    rjmp PCINT0 ; PCINT0 Handler
    rjmp PCINT1 ; PCINT1 Handler
    rjmp PCINT2 ; PCINT2 Handler
```

Interrupt Initialization

Interrupt Initialization

The *sequence* of initializing interrupts is important.

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.
3. enable (various registers)

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.
3. enable (various registers)

   for each individual source required

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.
3. enable (various registers)

   for each individual source required
4. **global** enable *all* interrupts (**STATUS**)

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.
3. enable (various registers)

   for each individual source required
4. **global** enable *all* interrupts (**STATUS**)

   after all individual sources have been initialized

Interrupt Initialization

The *sequence* of initializing interrupts is important.

1. configure individual interrupt sources (various registers)

   any needed configuration, initialization of variables, etc.
2. clear flags (various registers)

   On reset, many interrupt flags are in an unknown state, so interrupts may be incorrectly identified as pending.
3. enable (various registers)

   for each individual source required
4. **global** enable *all* interrupts (**STATUS**)

   after all individual sources have been initialized

   needed for *any* interrupt service to occur

Example: Timer 0

Example: Timer 0

1. configure individual interrupt sources (various registers)

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

   write '1' to clear

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

   write '1' to clear
3. enable (various registers)

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

   write '1' to clear
3. enable (various registers)

   OCIE0A, OCIE0B, TOIE0 are in TIFR0

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

   write '1' to clear
3. enable (various registers)

   OCIE0A, OCIE0B, TOIE0 are in TIFR0

   for each individual source required

Example: Timer 0

1. configure individual interrupt sources (various registers)
2. clear flags (various registers)

   OCF0A, OCF0B, TOV0 are in TIFR0

   write '1' to clear
3. enable (various registers)

   OCIE0A, OCIE0B, TOIE0 are in TIFR0

   for each individual source required

   I is in STATUS

Interrupt Service Routines

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

   STATUS

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

   STATUS
2. confirm source

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

   STATUS

2. confirm source

   Check specific flag bit to confirm that the expected source caused the interrupt.

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

   STATUS

2. confirm source

   Check specific flag bit to confirm that the expected source caused the interrupt.

3. process

Interrupt Service Routines

The sequence of actions in *servicing* interrupts is also important.

1. save vital registers

   STATUS

2. confirm source

   Check specific flag bit to confirm that the expected source caused the interrupt.

3. process

   more on this later

Interrupt Service Routines (continued)

Interrupt Service Routines (continued)

3. process

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

   Some flags need to be cleared explicitly; others will happen as part of normal service.

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

   Some flags need to be cleared explicitly; others will happen as part of normal service.

   Always check whether a flag will be cleared automatically or not.

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

   Some flags need to be cleared explicitly; others will happen as part of normal service.

   Always check whether a flag will be cleared automatically or not.
5. restore vital registers

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

   Some flags need to be cleared explicitly; others will happen as part of normal service.

   Always check whether a flag will be cleared automatically or not.
5. restore vital registers

   STATUS

Interrupt Service Routines (continued)

3. process
4. clear flags (if required)

   Some flags need to be cleared explicitly; others will happen as part of normal service.

   Always check whether a flag will be cleared automatically or not.
5. restore vital registers

   STATUS
6. **reti**

   like a subroutine return, but re-enables interrupts

Rules for Interrupts

Rules for Interrupts

- Never wait in interrupts.

Rules for Interrupts

- Never wait in interrupts.

  set flags, have waiting in main program

Rules for Interrupts

- Never wait in interrupts.

  set flags, have waiting in main program
- Use interrupts for events of very short duration

Rules for Interrupts

- Never wait in interrupts.

  set flags, have waiting in main program
- Use interrupts for events of very short duration

  i.e. if polling may miss them entirely

Interrupt Sources

Interrupt Sources

internal, external

Interrupt Sources

internal, external

pin change, timers, serial, ADC, etc.