Overview

- AVR Interrupts
- Interrupt Vector Table
- System Reset
- Watchdog Timer
- Timer/Counter0
- Interrupt Service Routines

Interrupts in AVR

- The number of interrupts varies with specific AVR device.
- Two types of interrupts: Internal interrupts and external interrupts.
  - Internal interrupts: Generated by on-chip I/O devices.
  - External interrupts: Generated by external I/O devices.
- For most internal interrupts, they don’t have an individual enable/disable bit.
  - Program cannot enable/disable these interrupts.
- External interrupts have an individual enable/disable bit.
  - Program can enable/disable these interrupts.
  - An external interrupt can be rising edge-triggered, or falling edge-triggered or low level-triggered.
    - Special I/O registers (External Interrupt Control Registers EICRA and EICRB in Mega64) to specify how each external interrupt is triggered.
Interrupts in AVR (Cont.)

- There is a global interrupt enable/disable bit, the I-bit, in Program Status Register SREG.
  - Setting the I-bit will enable all interrupts except those with individual enable/disable bit. Those interrupts are enabled only if both I and their own enable/disable bit are set.
  - The I-bit is cleared after an interrupt has occurred and is set by the instruction RETI.
  - Programmers can use SEI and CLI to set and clear the I-bit.
  - If the I-bit is enabled in the interrupt service routine, nested interrupts are allowed.
- SREG is not automatically saved by hardware when entering an interrupt service routine.
  - An interrupt service routine needs to save it and other conflict registers on the stack at the beginning and restore them at the end.
- Reset is handled as a nonmaskable interrupt.
- Each interrupt has a 4-byte interrupt vector, containing an instruction to be executed after MCU has accepted the interrupt.
- Each interrupt vector has a vector number, an integer from 1 to n, the maximum number of interrupts.
  - The priority of each interrupt is determined by its vector number.
    - The lower the vector number, the higher priority.
  - All interrupt vectors, called Interrupt Vector Table, are stored in a contiguous section in flash memory.
    - Starts from 0 by default.
    - Can be relocated.

### Interrupt Vectors in Mega64

<table>
<thead>
<tr>
<th>Vector No.</th>
<th>Program Address</th>
<th>Source</th>
<th>Interrupt Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x0000</td>
<td>RESET</td>
<td>External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset</td>
</tr>
<tr>
<td>2</td>
<td>0x0002</td>
<td>INT0</td>
<td>External Interrupt Request 0</td>
</tr>
<tr>
<td>3</td>
<td>0x0004</td>
<td>INT1</td>
<td>External Interrupt Request 1</td>
</tr>
<tr>
<td>4</td>
<td>0x0006</td>
<td>INT2</td>
<td>External Interrupt Request 2</td>
</tr>
<tr>
<td>5</td>
<td>0x0008</td>
<td>INT3</td>
<td>External Interrupt Request 3</td>
</tr>
<tr>
<td>6</td>
<td>0x000A</td>
<td>INT4</td>
<td>External Interrupt Request 4</td>
</tr>
<tr>
<td>7</td>
<td>0x000C</td>
<td>INT5</td>
<td>External Interrupt Request 5</td>
</tr>
<tr>
<td>8</td>
<td>0x000E</td>
<td>INT6</td>
<td>External Interrupt Request 6</td>
</tr>
<tr>
<td>9</td>
<td>0x0010</td>
<td>INT7</td>
<td>External Interrupt Request 7</td>
</tr>
<tr>
<td>10</td>
<td>0x0012</td>
<td>TIMER2 COMP</td>
<td>Timer/Counter2 Compare Match</td>
</tr>
<tr>
<td>11</td>
<td>0x0014</td>
<td>TIMER2 OVF</td>
<td>Timer/Counter2 Overflow</td>
</tr>
<tr>
<td>12</td>
<td>0x0016</td>
<td>TIMER1 CAPT</td>
<td>Timer/Counter1 Capture Event</td>
</tr>
<tr>
<td>13</td>
<td>0x0018</td>
<td>TIMER1 COMPB</td>
<td>Timer/Counter1 Capture Match A</td>
</tr>
<tr>
<td>14</td>
<td>0x001A</td>
<td>TIMER1 COMPC</td>
<td>Timer/Counter1 Capture Match B</td>
</tr>
<tr>
<td>15</td>
<td>0x001C</td>
<td>TIMER1 OVF</td>
<td>Timer/Counter1 Overflow</td>
</tr>
<tr>
<td>16</td>
<td>0x001E</td>
<td>TIMER0 COMP</td>
<td>Timer/Counter0 Compare Match</td>
</tr>
<tr>
<td>17</td>
<td>0x0020</td>
<td>TIMER0 OVF</td>
<td>Timer/Counter0 Overflow</td>
</tr>
<tr>
<td>18</td>
<td>0x0022</td>
<td>SPI, STC</td>
<td>SPI Serial Transfer Complete</td>
</tr>
<tr>
<td>19</td>
<td>0x0024</td>
<td>USART0, RX</td>
<td>USART0, Rx Complete</td>
</tr>
<tr>
<td>20</td>
<td>0x0026</td>
<td>USART0, UDRE</td>
<td>USART0 Data Register Empty</td>
</tr>
<tr>
<td>21</td>
<td>0x002E</td>
<td>USART0, TX</td>
<td>USART0, Tx Complete</td>
</tr>
<tr>
<td>22</td>
<td>0x0030</td>
<td>ADC</td>
<td>ADC Conversion Complete</td>
</tr>
<tr>
<td>23</td>
<td>0x0032</td>
<td>EE READY</td>
<td>EEPCON Ready</td>
</tr>
<tr>
<td>24</td>
<td>0x0034</td>
<td>ANALOG COMP</td>
<td>Analog Comparator</td>
</tr>
<tr>
<td>25</td>
<td>0x0036</td>
<td>TIMER1 COMPC</td>
<td>Timer/Counter1 Compare Match C</td>
</tr>
</tbody>
</table>
Interrupt Vectors in Mega64 (Cont.)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>0x0030H</td>
<td>TIMERS COMPA Timer/Counter3 Compare Match A</td>
</tr>
<tr>
<td>28</td>
<td>0x0031H</td>
<td>TIMERS COMPB Timer/Counter3 Compare Match B</td>
</tr>
<tr>
<td>29</td>
<td>0x0032H</td>
<td>TIMERS COMPC Timer/Counter3 Compare Match C</td>
</tr>
<tr>
<td>30</td>
<td>0x0033H</td>
<td>TIMERS CUF Timer/Counter3 Overflow</td>
</tr>
<tr>
<td>32</td>
<td>0x003CH</td>
<td>USART1, UDRE USART1 Data Register Empty</td>
</tr>
<tr>
<td>33</td>
<td>0x003DH</td>
<td>USART1, TX USART1, Tx Complete</td>
</tr>
<tr>
<td>34</td>
<td>0x003EH</td>
<td>TWI Two-wire Serial Interface</td>
</tr>
<tr>
<td>35</td>
<td>0x0040H</td>
<td>SPM READY Store Program Memory Ready</td>
</tr>
</tbody>
</table>

Initialization of Interrupt Vector Table in Mega64

- Typically an interrupt vector contains a branch instruction (JMP or RJMP) that branches to the first instruction of the interrupt service routine.
- Or simply RETI (return-from-interrupt) if you don’t handle this interrupt.

Example of IVT Initialization in Mega64

```
.include "m64def.inc"
.cseg
.org 0
rjmp RESET ; Jump to the start of Reset interrupt service routine
;jmp IRC0 ; Relative jump is used assuming RESET is not far
        ; Long jump is used assuming IRQ0 is very far away
reti ; Return to the break point (No handling for this interrupt).
...
RESET: ; The interrupt service routine for RESET starts here.
...
IRQ0: ; The interrupt service routine for IRQ0 starts here.
```

RESET in Mega64

The ATmega64 has five sources of reset:

- Power-on Reset.
  - The MCU is reset when the supply voltage is below the Power-on Reset threshold (VPOT).
- External Reset.
  - The MCU is reset when a low level is present on the RESET pin for longer than the minimum pulse length.
- Watchdog Reset.
  - The MCU is reset when the Watchdog Timer period expires and the Watchdog is enabled.
RESET in Mega64 (Cont.)

• Brown-out Reset.
  - The MCU is reset when the supply voltage VCC is below the Brown-out Reset threshold (VBOT) and the Brown-out Detector is enabled.

• JTAG AVR Reset.
  - The MCU is reset as long as there is a logic one in the Reset Register, one of the scan chains of the JTAG system.

For each reset, there is a flag (bit) in MCU Control Register MCUCSR.

• These bits are used to determine the source of the RESET interrupt.

Watchdog Timer

• Used to detect software crash.

• Can be enabled or disabled by properly updating WDCE bit and WDE bit in Watchdog Timer Control Register WDTCR.

• 8 different periods determined by WDP2, WDP1 and WDP0 bits in WDTCR.

• If enabled, it generates a Watchdog Reset interrupt when its period expires.

• So program needs to reset it before its period expires by executing instruction WDR.

• When its period expires, Watchdog Reset Flag WDRF in MCU Control Register MCUCSR is set.

  - This flag is used to determine if the watchdog timer has generated a RESET interrupt.
Timer Interrupt

Timer interrupt has many applications:

- Used to schedule (real-time) tasks
  - Round-Robin scheduling
    - All tasks take turn to execute for some fixed period.
  - Real-time scheduling
    - Some tasks must be started at a particular time and finished by a deadline.
    - Some tasks must be periodically executed.
- Used to implement a clock
  - How much time has passed since the system started?

Timer0 in AVR

8-bit timer with the following features:

- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 10-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV0 and OCF0)
  - It generates a Timer0 Overflow Interrupt Timer0OVF when it overflows.
  - It generates a Timer/Counter0 Output Match Interrupt Timer0COMP when the timer/counter value matches the value in Output Compare Register OCR0.
  - Timer0OVF and Timer0COMP can be individually enabled/disabled.
- Allows Clocking from External 32 kHz Watch Crystal Independent of the I/O Clock

Timer Interrupt (Cont.)

- Used to synchronize tasks.
  - Task A can be started only if a certain amount of time has passed since the completion of task B.
- Can be coupled with wave-form generator to support Pulse-Width Modulation (PWM).
  - Details to be covered later.

Timer0 In AVR–Block Diagram
Prescaler for Timer0

Timer0 Registers

Seven I/O registers for Timer0:

- Timer/Counter Register TCNT0.
  - Contains the current timer/counter value.

- Output Compare Register OCR0.
  - Contains an 8-bit value that is continuously compared with the counter value (TCNT0).

- Timer/Counter Control Register TCCR.
  - Contains control bits.

- Timer/Counter Interrupt Mask Register TIMSK (shared with other timers).
  - Contains enable/disable bits.

Timer0 Registers (Cont.)

- Timer/Counter Interrupt Flag Register TIFR (shared with other timers).
  - Contains interrupt flags.

- Asynchronous Status Register ASSR.
  - Contains control bits for asynchronous operations.

- Special Function I/O Register SFIOR.
  - Contains synchronization mode bit and prescaler reset bit.

Timer Control Register
Timer Control Register (Cont.)

- The mode of operation, i.e., the behavior of the Timer/Counter and the Output Compare pins, is defined by the combination of the Waveform Generation mode (WGM01:0) and Compare Output mode (COM01:0) bits.
- The simplest mode of operation is the Normal Mode (WGM01:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP = 0xFF) and then restarts from the bottom (0x00).
- Refer to Mega64 Data Sheet (pages 96~100) for details.

Timer/Counter Interrupt Mask Register

- Bit 1 – OCIE0: Timer/Counter0 Output Compare Match Interrupt Enable.
  - 1: Enabled  0: Disabled
- Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable.
  - 1: Enabled  0: Disabled

ISR Example

.include "m64def.inc" ; This program implements a second counter
            ; using Timer0 interrupt.
.def temp r16
.MACRO Clear
  ldi r28, low(@0) ; Load the low byte of label @0
  ldi r29, high(@0) ; Load the high byte of label @0
  clr temp
  st y, temp
  st –y, temp ; Initialize the two-byte integer at @0 to 0
.ENDMACRO

ISR Example

.dseg
SecondCounter: ; Two-byte second counter.
.byte 2
.TempCounter: ; Temporary counter. Used to determine
            ; if one second has passed
.cseg
.org 0
  jmp DEFAULT ; No handling for IRQ0.
  jmp DEFAULT ; No handling for IRQ1.
ISR Example (Cont.)

...  
jmp Timer0 ; Jump to the interrupt handler for Timer 0 overflow.
jmp DEFAULT ; No handling for all other interrupts.
...
DEFAULT: reti ; No handling for this interrupt

RESET: ldi temp, high(RAMEND) ; Initialize stack pointer
   out SPH, temp
   ldi temp, low(RAMEND)
   out SPL, temp
   ... ; Insert further initialization code here
   rjmp main

Timer0 ISR

Timer0: push SREG ; Prologue starts.
   push r29
   push r28
   push r25
   push r24 ; Prologue ends.
ldi r28, low(TempCounter) ; Load the address of the temporary
ldi r29, high(TempCounter) ; counter.
ld r24, y+ ; Load the value of the temporary counter.
ld r25, y
adiw r25:r24, 1 ; Increase the temporary counter by one.

Timer0 ISR (Cont.)

cpi r24, low(3597) ; Check if (r25:r24)=3597
ldi temp, high(3597) ; 3597 = 106/278
cpc r25, temp
bne NotSecond
clr temp ; One second has passed since last interrupt
st y, temp ; Reset the temporary counter.
st –y, temp
ldi r30, low(SecondCounter) ; Load the address of the second
ldi r31, high(SecondCounter) ; counter.
ld r24, z+ ; Load the value of the second counter.
ld r25, z
adiw r25:r24, 1 ; Increase the second counter by one.

Timer0 ISR (Cont.)

st z, r25 ; Store the value of the second counter.
st –z, r24

NotSecond:
st y, r25 ; Store the value of the temporary counter.
st –y, r24
pop r25 ; Epilogue starts;
pop r24 ; Restore all conflict registers from the stack.
pop r28
pop r29
pop SREG
reti ; Return from the interrupt.
ISR Example (Cont.)

main:
  Clear TempCounter ; Initialize the temporary counter to 0
  Clear SecondCounter ; Initialize the second counter to 0
  ldi temp, 0b00000010
  out TCCR0, temp ; Prescaling value=8 → 256*8/7.3728
  ldi temp, 1<<TOIE0 ; =278 microseconds
  out TIMSK, temp ; T/C0 interrupt enable
  sei ; Enable global interrupt
  loop: jmp loop ; loop forever

Comments: 1: The frequency of Timer clock in Mega64 is 7.3728Mhz.
     2: Prescaling value is set to 8. Since the maximum value of a 8-bit counter is 255. Timer0 Overflow Interrupt occurs every 256*8/7.3728 =278 microseconds.

Non-Nested Interrupts

• Interrupt Service Routines cannot be interrupted by another interrupt.

Nested Interrupts

• Interrupt Service Routines can be interrupted by another interrupt.

Reading

Read the following sections in Mega64 Data Sheet.
1. Overview
2. AVR CPU Core
3. System Control and Reset.
4. Watchdog Timer.
5. Interrupts.
7. 8-bit Time/Counter0 with PWM and Asynchronous Operation.