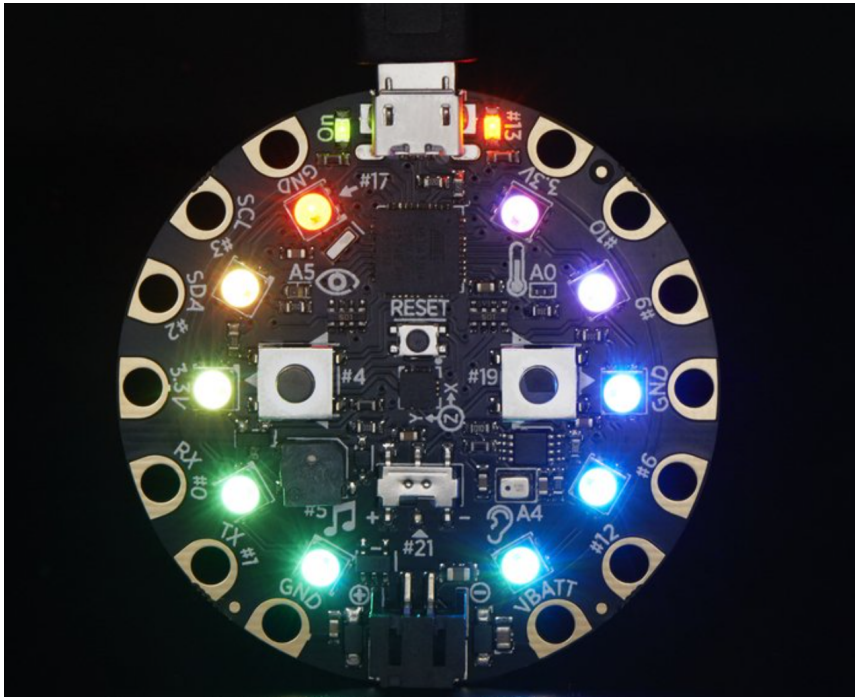
INTERRUPTS

EE 4144 Real Time Embedded Systems



Interrupts

- Allow program to respond to events when they occur
- Allow program to ignore events until the occur
- External events e.g.:
  - UART ready with/for next character
  - Signal change on pin
  - Action depends on context
  - # of edges arrived on pin
- Internal events e.g.:
  - Power failure
  - Arithmetic exception
  - Timer “tick”

ATmega328 Interrupts (Example)

VectorNo.	Program Address[2]	Source	Interrupt Definition
1	0x0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B

ATmega328 Interrupts (cont)

VectorNo.	Program Address(2)	Source	Interrupt Definition
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Interrupt Model

- When an interrupt event occurs:
  - Processor does an automatic procedure call
  - CALL automatically done to address for that interrupt
    - Push current PC, Jump to interrupt address
  - Each event has its own interrupt address
  - The global interrupt enable bit (in SREG) is automatically cleared
    - i.e. nested interrupts are disabled
    - SREG bit can be set to enable nested interrupts if desired
- Interrupt procedure, aka “interrupt handler”
  - Does whatever it needs to, then returns via RETI
  - The global interrupt enable bit is automatically set on RETI
  - One program instruction is always executed after RETI

Interrupts

- Type 1 – Event is remembered when interrupt is disabled
  - If interrupt is not enabled, flag is set
    - When interrupt is enabled again, interrupt takes place,and flag is reset
- Type 2 – Event is not remembered when interrupt is disabled
  - Signal level causes interrupt
  - If level occurs when interrupt is enabled, interrupt takes place
  - If interrupt is not enabled, and level goes away before the interrupt is enabled, nothing happens

Interrupt Model

- Interrupt hander is invisible to program
  - Except through side-effects, e. g. via flags or variables
  - Changes program timing
    - Can’t rely on “dead-reckoning” using instruction timing
- Must be written so they are invisible
  - Cannot stomp on program state, e. g. registers
  - Save and restore any registers used
    - Including SREG

Interrupt Vectors

- Table in memory containing the first instruction of each interrupt handler
- Typically at program address 0

```
Address  Labels Code                 Comments
0x0000          jmp    RESET         ; Reset Handler
0x0002          jmp    EXT_INT0      ; IRQ0 Handler
0x0004          jmp    EXT_INT1      ; IRQ1 Handler
0x0006          jmp    PCINT0        ; PCINT0 Handler
0x0008          jmp    PCINT1        ; PCINT1 Handler
0x000A          jmp    PCINT2        ; PCINT2 Handler
0x000C          jmp    WDT           ; Watchdog Timer Handler
0x000E          jmp    TIM2_COMPA    ; Timer2 Compare A Handler
0x0010          jmp    TIM2_COMPB    ; Timer2 Compare B Handler
0x0012          jmp    TIM2_OVF      ; Timer2 Overflow Handler
0x0014          jmp    TIM1_CAPT     ; Timer1 Capture Handler
0x0016          jmp    TIM1_COMPA    ; Timer1 Compare A Handler
0x0018          jmp    TIM1_COMPB    ; Timer1 Compare B Handler
0x001A          jmp    TIM1_OVF      ; Timer1 Overflow Handler
0x001C          jmp    TIM0_COMPA    ; Timer0 Compare A Handler
0x001E          jmp    TIM0_COMPB    ; Timer0 Compare B Handler
```

Interrupt Vectors

- If interrupts are not used, this memory can be used as part of the program
  - i.e. nothing special about this part of memory
- Example interrupt routine
  - RESET: Sets up the stack pointer

```
0x0033RESET:    ldi    r16, high(RAMEND); Main program start
0x0034          out    SPH,r16          ; Set Stack Pointer to top of RAM
0x0035          ldi    r16, low(RAMEND)
0x0036          out    SPL,r16
0x0037          sei                     ; Enable interrupts
0x0038          <instr>  xxx
```

Defined ISR's

```
#define INT0_vect _VECTOR(1) /* External Interrupt Request 0 */
#define INT1_vect _VECTOR(2) /* External Interrupt Request 1 */
#define PCINT0_vect _VECTOR(3) /* Pin Change Interrupt Request 0 */
#define PCINT1_vect _VECTOR(4) /* Pin Change Interrupt Request 0 */
#define PCINT2_vect _VECTOR(5) /* Pin Change Interrupt Request 1 */
#define WDT_vect _VECTOR(6) /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect _VECTOR(7) /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect _VECTOR(8) /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect _VECTOR(9) /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect _VECTOR(10) /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect _VECTOR(11) /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect _VECTOR(12) /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect _VECTOR(13) /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect _VECTOR(14) /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect _VECTOR(15) /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect _VECTOR(16) /* Timer/Couner0 Overflow */
#define SPI_STC_vect _VECTOR(17) /* SPI Serial Transfer Complete */
#define USART_RX_vect _VECTOR(18) /* USART Rx Complete */
#define USART_UDRE_vect _VECTOR(19) /* USART, Data Register Empty */
#define USART_TX_vect _VECTOR(20) /* USART Tx Complete */
#define ADC_vect _VECTOR(21) /* ADC Conversion Complete */
#define EE_READY_vect _VECTOR(22) /* EEPROM Ready */
#define ANALOG_COMP_vect _VECTOR(23) /* Analog Comparator */
#define TWI_vect _VECTOR(24) /* Two-wire Serial Interface */
#define SPM_READY_vect _VECTOR(25) /* Store Program Memory Read */
```

Interrupts

- Global interrupt enable
  - Bit in SREG
  - Allows all interrupts to be disabled with one bit
  - sei() – set the bit
  - cli() – clear the bit
- Interrupt priority is determined by order in table
  - Lower addresses have higher priority
- ISR(vector) – Interrupt routine definition
- reti() – return from interrupt
  - automatically generated for ISR

External Interrupts

- Monitors changes in signals on pins
- What causes an interrupt can be configured
  - by setting control registers appropriately
- Pins:
  - INT0 and INT1 – range of event options
    - INT0 – PORT D [2]
    - INT1 – PORT D [3]
  - PCINT[23:0] – any signal change (toggle)
    - PCINT[7:0] – PORT B [7:0]
    - PCINT[14:8] – PORT C [6:0]
    - PCINT[23:16] – PORT D [7:0]
- Pulses on inputs must be slower than I/O clock rate

INT0 and INT1

- External Interrupt Control Register:

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Sense Control (INT0 is the same)

Table 12-1. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

INT0 and INT1

- External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - If INT# bit is set (and the SREG I-bit is set), then interrupts are enabled on pin INT#

- External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - Interrupt flag bit is set when a change triggers an interrupt request
  - Flag is cleared automatically when interrupt routine is executed
  - Flag can be cleared by writing a 1 to it

Arduino Language Support for External Interrupts

- attachInterrupt(interrupt, function, mode)
  - interrupt: 0 or 1
  - function: interrupt function to call
  - mode: LOW, CHANGE, RISING, FALLING
- detachInterrupt(interrupt)
- interrupts( ) – Enable interrupts : sei( )
- noInterrupts( ) – Disable interrupts : cli( )

PCINT[23:0]

- Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - PCIE2 enables interrupts for PCINT[23:16]
  - PCIE1 enables interrupts for PCINT[14:8]
  - PCIE0 enables interrupts for PCINT[7:0]
- Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - PCIF# set if corresponding pins generate an interrupt request
  - Cleared automatically when interrupt routine is executed

PCINT[23:0]

- Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Each bit controls whether interrupts are enabled for the corresponding pin
- Change on any enabled pin causes an interrupt
- (Mask registers 1 and 0 are similar)

Timer/Counter 0 Interrupts

- Timer/Counter 0 Interrupt Mask

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - TOIE0 – Timer Overflow interrupt
  - OCIE0A/B – Compare A/B interrupt
- Timer/Counter 1 Interrupt Flags

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - TOV0 – Timer Overflow flag
  - OCF0A/B – Compare A/B interrupt flag

External and Pin Change Interrupts

```
#define PCIFR _SFR_IO8(0x1B)
#define PCIF0 0
#define PCIF1 1
#define PCIF2 2
#define EIFR _SFR_IO8(0x1C)
#define INTF0 0
#define INTF1 1
#define EIMSK _SFR_IO8(0x1D)
#define INT0 0
#define INT1 1
#define PCICR _SFR_MEM8(0x68)
#define PCIE0 0
#define PCIE1 1
#define PCIE2 2
#define EICRA _SFR_MEM8(0x69)
#define ISC00 0
#define ISC01 1
#define ISC10 2
#define ISC11 3
```

```
#define PCMSK0 _SFR_MEM8(0x6B)
#define PCINT0 0
...
#define PCINT7 7
#define PCMSK1 _SFR_MEM8(0x6C)
#define PCINT8 0
...
#define PCINT14 6
#define PCMSK2 _SFR_MEM8(0x6D)
#define PCINT16 0
...
#define PCINT23 7
```

Timer/Counter Interrupts

```
#define TIFR0 _SFR_IO8(0x15)
#define TOV0 0
#define OCF0A 1
#define OCF0B 2
#define TIFR1 _SFR_IO8(0x16)
#define TOV1 0
#define OCF1A 1
#define OCF1B 2
#define ICF1 5
#define TIFR2 _SFR_IO8(0x17)
#define TOV2 0
#define OCF2A 1
#define OCF2B 2
```

```
#define TIMSK0 _SFR_MEM8(0x6E)
#define TOIE0 0
#define OCIE0A 1
#define OCIE0B 2
#define TIMSK1 _SFR_MEM8(0x6F)
#define TOIE1 0
#define OCIE1A 1
#define OCIE1B 2
#define ICIE1 5
#define TIMSK2 _SFR_MEM8(0x70)
#define TOIE2 0
#define OCIE2A 1
#define OCIE2B 2
```

Timer Interrupt Program Example

```
void setup() {
DDRB =                  ;               // Pin 13 as output
// Using timer 2
// Set to Normal mode, Pin OC0A disconnected
TCCR2A =                ;
// Prescale clock by 1024
// Interrupt every 256K/16M sec = 1/64 sec
TCCR2B =                ;
// Turn on timer overflow interrupt flag
TIMSK2 = ;
// Turn on global interrupts
sei();
}
char timer = 0;

ISR(          _vect) {
timer++;
PORTB =  ;
}

void loop()
{
// Nothing to do
}
```

Timer Example #2

```
void setup() {
DDRB =                  ;               // Pin 13 OUTPUT
// Using timer 2
// Set to CTC mode, Pin OC0A disconnected
TCCR2A =                ;
// Prescale clock by 1 (no prescale)
TCCR2B =                ;

// Set compare register
OCR2A =  ;
// Turn on timer compare A interrupt flag
TIMSK2 = ;
// Turn on global interrupts
sei();
}
char timer = 0;

ISR(          _vect) {
timer++;
PORTB =  ;
}
void loop()
{
// Nothing to do
}
```

External Interrupt Example

```
#define pinint0
#define pinint1
void setup() {
pinMode(pinint0,     );
pinMode(pinint1,    );
Serial.begin(9600);
// External interrupts 0 and 1
// Interrupt on rising edge
EICRA = ;
// Enable both interrupts
EIMSK = ;
// Turn on global interrupts
sei();
}
ISR(        _vect) {
}
ISR(         _vect) {
}
```

```
// Print out the information
void loop()
{
Serial.print("X: ");
Serial.print(percent0);
Serial.print(" Y: ");
Serial.println(percent1);
}
```