Interrupts and External Interrupts on the AVR

ME-ECE 4370/5370

Introduction:

Interrupts provide hardware triggering of events that over-ride the normal software program flow. These interrupts can perform some combination of hardware functions and software-defined functions (defined in an interrupt service/handling routine or function). The ATmega series has interrupts for reset, External interrupts (8), Pin Change Interrupts (24), timer counter matches and overflows (timer counters 0, 1, 2,3,4,5), SPI, USART0,1,2,3, Analog comparator, ADC complete and a few others.

The external interrupts are triggered by an input state change or at a low level.

Interrupts are extremely useful for developing advanced applications. Examples include counting events, timing events, decoding waveforms, etc. This set of notes will focus on the external interrupts, INT and PCINT.

Interrupt Handling:

Interrupts have a global and local enable bits. The global interrupt enable bit is in the Status Register and set through the function sei. The local enable bits are set in corresponding registers. The first 256 bytes of memory (approximately) are used for the interrupt jump vectors. When an interrupt is enabled and occurs, an interrupt flag is set (in most cases). This causes the program counter to fetch the memory address stored at the corresponding jump vector location to execute the interrupt service routine or interrupt handling routine. The hardware then clears this flag. The flag can also be cleared in software. When an interrupt is generated, all other interrupts are temporarily disabled until the interrupt is complete (this is standard operation to avoid interrupting and interrupt, this can be overwritten).

There are a set of interrupts that do not set a corresponding interrupt flag. These are not discussed in this set of notes.

External Interrupts

The external interrupts are triggered on the INT7:0 pins (Interrupt pins) and the PCINT23:0 pins (Pin Chang INTerrupt pins). The 8 INT pins are full featured to occur on rising, falling, toggle edges or low state with ISR (Interrupt Service Routines) dedicated to each. The PCINT pins are lower featured, occur on toggle state, and are grouped into three ISRs: one for PCINT7:0, one for PCINT15:8, and one for PCINT23:16.

The remainder of this document proceeds as follows:

1) Summary table of ports and eternal interrupts
2) Summary of important registers for External Interrupts
3) Details of registers for External Interrupts
4) Interrupt Vector table (for all interrupts)
5) Software Checklist for using External Interrupts
6) Example program for External Interrupts

1. Summary of important registers for Input Capture/Output Compare

EICRA, EICRB, External Interrupt Control Register A, B: This 8-bit register assigns the interrupt sense control

EIMSK: External Interrupts Mask: Locally enables the External Interrupt pins

EIFR: External Interrupt Flag Register: Contains the flags that get set when an interrupt occurs (when locally and globably enabled).

PCICR Pin Change Interrupt Control Register: Locally enables the Pin Change Interrupt pins in groups of 8

PCIFR Pin Change Interrupt Flag Register: Contains the flags that get set when an interrupt occurs (when locally and globably enabled).

PCMSK2,1,0 Pin Change Mask Register 2,1, 0: Provides masks to determine whether an input on the corresponding pin contributes to generating an interrupt.

2. Details of registers for External Interrupts

External Interrupt Table:

Ext. Int	Port	Arduino pin	Ext. Int	Port	Arduino pin	Ext. Int	Port	Arduino pin
INT0	PD0	21	PCINT0	PB0	53	PCINT8	PE0	0
INT1	PD1	20	PCINT1	PB1	52	PCINT9	PJ0	15
INT2	PD2	19	PCINT2	PB2	51	PCINT10	PJ1	14
INT3	PD3	18	PCINT3	PB3	50	PCINT11		
INT4	PE4	2	PCINT4	PB4	10	PCINT12		
INT5	PE5	3	PCINT5	PB5	11	PCINT13		
INT6	PE6*	?	PCINT6	PB6	12	PCINT14		
INT7	PE7*	?	PCINT7	PB7	13	PCINT15		
						PCINT16	PK0	8
						PCINT17	PK1	9
						PCINT18	PK2	10
						PCINT19	PK3	11
						PCINT20	PK4	12
						PCINT21	PK5	13
						PCINT22	PK6	14
						PCINT23	PK7	15

REGISTERS FOR INT7:0

EICRA – External Interrupt Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x69)	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
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	

ISCn0, ISCn1: Sense control bits. The states define behavior of the interrupt according to table 15-1 below

EICRB – External Interrupt Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x6A)	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
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	

ISCn0, ISCn1: Sense control bits. The states define behavior of the interrupt according to table 15-1 below

Table 15-1. Interrupt Sense Control[1]

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request
0	1	Any edge of INTn generates asynchronously an interrupt request
1	0	The falling edge of INTn generates asynchronously an interrupt request
1	1	The rising edge of INTn generates asynchronously an interrupt request

Note: 1. n = 3, 2, 1or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

The timing characteristics for an external input are summarized on the table below.

Table 15-2. Asynchronous External Interrupt Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
t_{INT}	Minimum pulse width for asynchronous external interrupt			50		ns

EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
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	

INTn: Writing a one to this bit indicates the corresponding interrupt pin is enabled. Writing a zero to this bit disables the corresponding interrupt pin.
In addition, the I-bit in the status register must be set to one for any interrupt to occur.
*NOTE: when enabled, activity on any of these external interrupt pins will request an interrupt even if the pin is enabled as an output.

EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	IINTF0	EIFR
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	

INTFn = Flag is set to one when an interrupt occurs. Timer enable (1 enabled, 0 disabled). The flag is cleared when the ~~IHR~~ ISR is executed. The flag could also be cleared by writing a one to it.

REGISTERS FOR PCINT0:23

PCICR – 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: Setting the PCIE2 bit to one locally enables the corresponding Interrupts: PC23:16
PCIE1: Setting the PCIE1 bit to one locally enables the corresponding Interrupts: PC15:8
PCIE0: Setting the PCIE0 bit to one locally enables the corresponding Interrupts: PC7:0
When enabled, any change (toggle) on the PCINT pins will cause an interrupt. The corresponding interrupt vectors (in the vector jump table) are PCI2, PCI1, PCI0.

PCIFR – 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	

PCIF2, 1, 0 – these flags get set when the corresponding interrupt is enalbled and occurs. The flag is cleared when the interrupt handling routine is executed, or can be cleared by writing a one to it.

PCMSK2 – 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	

PCINT23:16 – these pins determine if the corresponding PCINT23:16 pin generates an interrupt on toggle. If the bit is set here, and enabled locally and globally, the pin generates an interrupt.

PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
(0x6C)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
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	

PCINT15:8 – these pins determine if the corresponding PCINT15:8 pin generates an interrupt on toggle. If the bit is set here, and enabled locally and globally, the pin generates an interrupt.

PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
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	

PCINT7:0 – these pins determine if the corresponding PCINT7:0 pin generates an interrupt on toggle. If the bit is set here, and enabled locally and globally, the pin generates an interrupt.

Interrupt Vector Table

Table 14-1. Reset and Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016[3]	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI, STC	SPI Serial Transfer Complete
26	$0032	USART0 RX	USART0 Rx Complete
27	$0034	USART0 UDRE	USART0 Data Register Empty
28	$0036	USART0 TX	USART0 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator
30	$003A	ADC	ADC Conversion Complete

Table 14-1. Reset and Interrupt Vectors (Continued)

Vector No.	Program Address[2]	Source	Interrupt Definition
31	$003C	EE READY	EEPROM Ready
32	$003E	TIMER3 CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3 COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3 COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3 COMPC	Timer/Counter3 Compare Match C
36	$0046	TIMER3 OVF	Timer/Counter3 Overflow
37	$0048	USART1 RX	USART1 Rx Complete
38	$004A	USART1 UDRE	USART1 Data Register Empty
39	$004C	USART1 TX	USART1 Tx Complete
40	$004E	TWI	2-wire Serial Interface
41	$0050	SPM READY	Store Program Memory Ready
42	$0052[3]	TIMER4 CAPT	Timer/Counter4 Capture Event
43	$0054	TIMER4 COMPA	Timer/Counter4 Compare Match A
44	$0056	TIMER4 COMPB	Timer/Counter4 Compare Match B
45	$0058	TIMER4 COMPC	Timer/Counter4 Compare Match C
46	$005A	TIMER4 OVF	Timer/Counter4 Overflow
47	$005C[3]	TIMER5 CAPT	Timer/Counter5 Capture Event
48	$005E	TIMER5 COMPA	Timer/Counter5 Compare Match A
49	$0060	TIMER5 COMPB	Timer/Counter5 Compare Match B
50	$0062	TIMER5 COMPC	Timer/Counter5 Compare Match C
51	$0064	TIMER5 OVF	Timer/Counter5 Overflow
52	$0066[3]	USART2 RX	USART2 Rx Complete
53	$0068[3]	USART2 UDRE	USART2 Data Register Empty
54	$006A[3]	USART2 TX	USART2 Tx Complete
55	$006C[3]	USART3 RX	USART3 Rx Complete
56	$006E[3]	USART3 UDRE	USART3 Data Register Empty
57	$0070[3]	USART3 TX	USART3 Tx Complete

Notes:
1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Memory Programming" on page 325.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.
3. Only available in ATmega640/1280/2560.

External Interrupt Software Checklist:

External Interrupts

1. ~~Clear interrupts during setup process: CLI~~

2. Set corresponding pins to inputs or outputs based on program design

3. Choose the Interrupt trigger type based on Sense (edge) – EICRA, EICRB

4. Enable the local interrupts: EIMSK

5. ~~Enable global interrupts: SEI~~

6. Construct the corresponding ISR (Interrupt Service Routine): ISR(INTn_vect){ }

7. Reset the External Interrupt Flag (EIFR) – this is also automatically done by hardware.

Pin Change Interrupts

1. ~~Clear interrupts during setup process: CLI~~

2. Set corresponding pins to inputs or outputs based on program design

3. Choose which pins that contribute to the interrupt through a mask – PCMSK2,1,0

4. Enable the local interrupts: PCICR

5. ~~Enable global interrupts: SEI~~

6. Construct the corresponding ISR (Interrupt Service Routine): ISR(PCINTn_vect){ }

7. Reset the Pin Change Interrupt Flag (PCIFR) – this is also automatically done by hardware.

Additional notes for the

4. External Interrupt Programming Example

```
/* Sample program to do basic counting of an event using the external
interrupts
/* Tristan Hill, Stephen Canfield, March 2016
/* arranged for Arduino IDE


// define global variables
volatile uint16 ctr_4=0, ctr_5=0; //counts on channels INT4 and INT5
// define the Interrupt Service Routine - ISR functions:
ISR(INT4_vect){
        ctr_4++;                //index counter
        EIFR |= 0b00010000;     // clear flag
}
ISR(INT5_vect){
        ctr_5++;                //index counter
        EIFR |= 0b00100000;     // clear flag
}

void setup() {
        //setup serial monitor
        Serial.begin(9600);
        While(! Serial);
        Serial.println("Hello World!");

        // setup interups
        cli(): // clear global interrupts during setup
        DDRE &= 0b00110000; // set PE4,5 to inputs, all others unchanged
        EICRB |= 0b00001111; // trigger on rising edge of INT4, INT5
        EICRB |= 0b00110000; // locally enable INT4, INT5
        EIMSK |= 0b00110000; // clear flag on INT4, INT5
        sei(); // globally enable interrupts
}

void loop() {
        Serial.print("ISR4: ");
        Serial.print(ctr_4,DEC);
        Serial.print("ISR5: ");
        Serial.print(ctr_5,DEC);
        Serial.print("\n");
        delay(100);
}
```