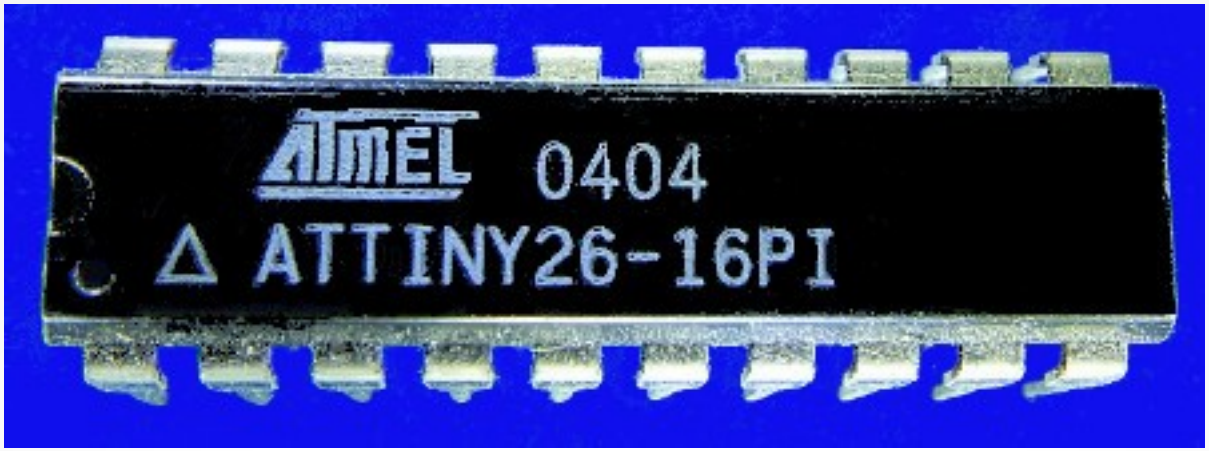


Lecture 9: External interrupts

Hardware, Internals and Programming of AVR Microcontrollers in Assembler

by
Gerhard Schmidt
Kastanienallee 20
D-64289 Darmstadt

External interrupts

- **All AVRs have pins where external level changes (either low to high, high to low or both) can generate an interrupt. These pins are named INTn, with n can be 0, 1 or any other type specific number.**
- **The type of level change, that generates an interrupt is controlled via the Interrupt Sense Control bits ISCn1:0. Those are located in the MCUCR port register or in larger devices with many INTn in a different port register.**

Table 9-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

- **The external interrupt is enabled by setting the INTn bit in GIMSK.**

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	–	INT0	PCIE1	PCIE0	–	–	–	–	GIMSK
Read/Write	R	R/W	R/W	R/W	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

PC interrupts

- **The INTn interrupts are on top of the interrupt vector jump list and have the highest priority.**
- **The ATtiny24 has only one external INT pin (PB2, pin 5 of the PDIP package), INT0.**
- **If you need more external interrupts: any pin can generate an interrupt. These interrupts are called PCINTn, in the ATtiny24 PCINT0 to PCINT7 are the interrupts generated on port pins PA and PCINT8 to PCINT11 on port pins PB.**
- **If enabled, any level change triggers a PCINT interrupt.**
- **To enable those external interrupts the respective PCINTn bits have to be set in the port registers PCMSK0 (PCINT0:7) or PCMSK1 (PCINT8:11). Any number of bits can be set or cleared.**
- **The interrupts can be enabled by setting the PCIE0 and/or PCIE1 bit in the GIMSK port register.**

PC interrupts, pin identification

- **Preferably only one of the PCINT0:7 and PCINT8:11 interrupts is enabled, so the attribution of the PCINT0 and PCINT1 to the pin is trivial.**
- **If more interrupts have to be utilized, the identification which of the pins has changed its level and triggered the PCINT0 or PCINT1 interrupt can use the Exclusive OR instruction EOR. EORing the current and the previous state of the port sets all bits that have changed.**

```
.EQU PcInt0Mask = 0b01010101 ; PA0/PA2/PA4/PA6 can cause the PCINT0
.DEF rPrev = R17 ; Previous port state
.DEF rCurr = R18 ; Current port state
; PCINT0 ISR
      IN R15, SREG ; Save SREG
      IN rCurr, PINA ; Read current port A state
      ANDI rCurr, 255 – PcInt0Mask ; Clear all bits that are not masked
      MOV R16, rCurr ; Copy current state
      EOR R16, rPrev ; Exclusive OR the previous state
```

Pin identification, continued

```
SBRC R16, 0 ; Skip next instruction if bit 0 is clear
RCALL Bit0Changed
SBRC R16, 2 ; Test bit 2 changed
RCALL Bit2Changed
SBRC R16, 4 ; Test bit 4 changed
RCALL Bit4 Changed
SBRC R16, 6 ; Test bit 6 changed
RCALL Bit6Changed
MOV rPrev, rCurr ; Copy current over previous
OUT SREG, R15 ; Restore SREG
RETI ; Return from interrupt
```

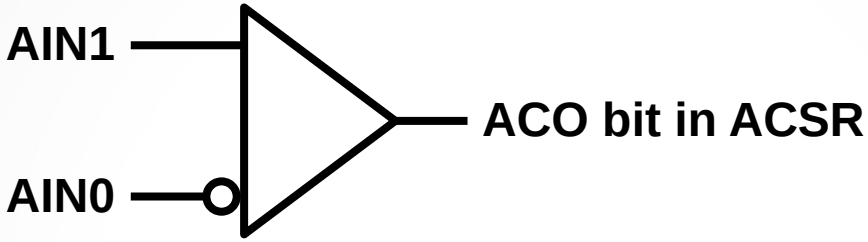
- **The subroutines BitNChanged can react specifically on every changed input pin.**
- **If the PCINT0/PCINT1 interrupts shall react on pressed keys connected to the pins, it is sufficient to react on a low level on these pins. Skip on Bit in I/o Set SBIS can be used, followed by those RCALLs.**

Suppression of key bouncing

- Keys tend to bounce, maybe not when new but after ageing. So it is a good idea
  - to clear and start a timer if one of the keys has been pressed,
  - to disable further key reactions within the following 20 or 50 ms,
  - to restart the timer by clearing its TCNT port register if another active key action happens in between,
  - to enable key reactions again when the timer reaches its end point.
- Blocking needs a flag bit here. Use any bit in any register, with Set Bit in Register SBR and Clear Bit in Register CBR to set and clear this flag.
- If not used for other purposes, the T flag in SREG can be set with SET and cleared with CLT. Note that saving the SREG in interrupt service routines overwrites the T flag when restoring its state.

The analog comparer

- **One fifth of all AVRs have analog comparers on board. If enabled (ACD bit in ACSR is one), the analog comparer compares the analog voltages on the pins AIN1 and AIN0 and set the ACO bit in ACSR according to the result.**



Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **If the ACIE bit is set, an interrupt can be generated. If the ACS1:0 bits are 0b00 this is done on every level change. The ACS bits can enable interrupts only on falling edges (0b10) or on rising edges (0b11).**

Serial communication

- **Half of the AVR devices have serial communication hardware on board.**
- **Two types of serial communication are supported:**
  - **Synchroneous interfaces such as the Two-Wire Interface (TWI) or I2C, see Universial Serial Communication (USC) in the data book,**
  - **Asynchroneous interfaces such as Universial Asynchroneous Receiver and Transmitter (UART).**
- **All serial interfaces can initiate diverse interrupt types.**
- **Use avr_sim („Project“, „New“, „Device-selector“) to find all AVR types that have such serial interfaces by selecting serial pins (SDA0 or UXD0).**

Questions and tasks in Lecture 9

Task 9-1: Write a program that follows any changes on the INT0 pin and lights a LED if the input is low.

Bonus question: What has to be changed if the LED lights on the input high?

Questions and tasks in Lecture 9 - Continued

Task 9-2: Write a program that counts the key bounces on PB0 and display those on four LEDs attached to PA0 to PA3.

Bonus task: Clear the display if the PCINT input pin has been inactive for longer than five seconds.

Questions and tasks in Lecture 9 - Continued

Task 9-3: Build this hardware and measure the frequency on the input with the analog comparer. If this is between 49 and 51 Hz, light the green LED.

Touch the f_{IN} connector to test the hard- and software.