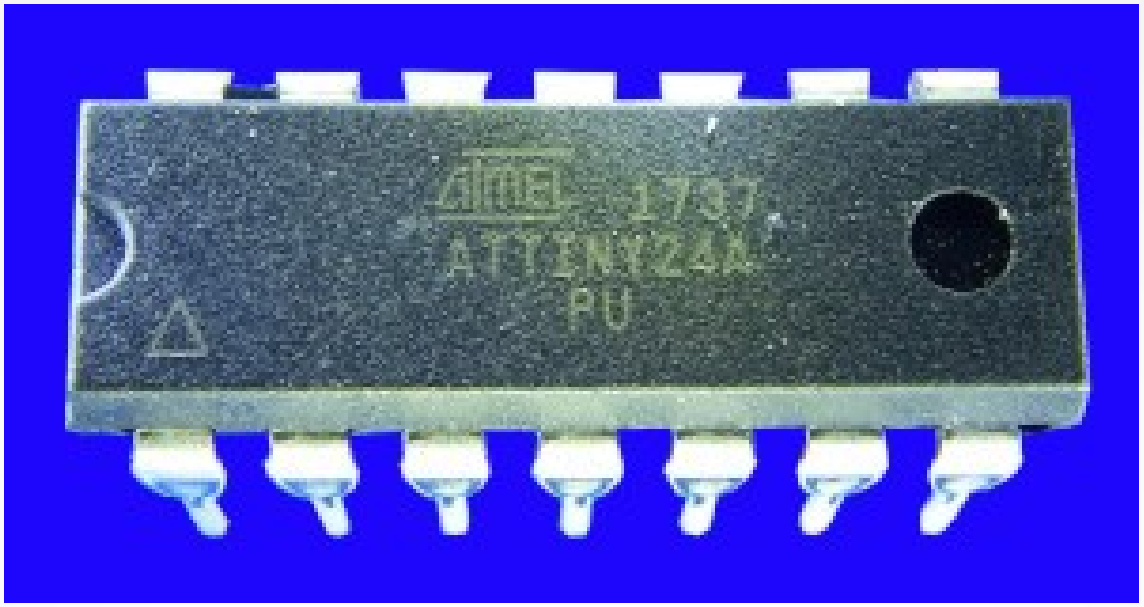


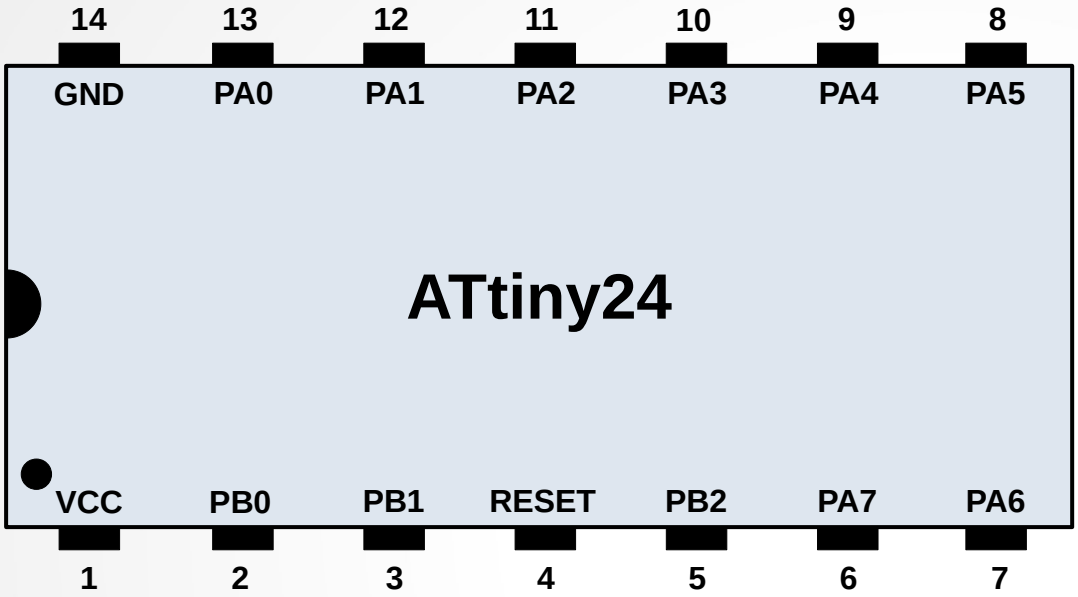
Lecture 2: Manipulating pins

Hardware, Internals and Programming of AVR Microcontrollers in Assembler

by
Gerhard Schmidt
Kastanienallee 20
D-64289 Darmstadt

How to manipulate pins

- **The pins of an ATtiny24 as seen from above:**



Pins 1 and 14 are for the supply voltage: VCC: +2.7 to +5.5 Volt, GND: 0 Volt

Pin 4: RESET, LOW stops the controller, LOW to HIGH restarts the program at address 0, should be tied with a resistor of 10 kΩ to VCC

- **The ATtiny24 has eight pins called port A: PA0 to PA7**
- **Additionally it has three pins of a port B: PB0 to PB2**
- **Each of those 11 pins can either be an input or an output pin**
- **If a pin is input, a Pull-Up resistor of 50 kΩ can be switched on to tie the pin to VCC (if no external connection pulls it LOW)**

Making a port to be output

- **Switching a port to be output is simple, just set its direction bit to ONE.**
- **The direction bits are located in a memory space called „port registers“, named data direction register „DDRA“ and „DDRB“ at the addresses 0x1A and 0x17:**

22. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C
0x3E (0x5E)	SPH	–	–	–	–	–	–	SP9	SP8
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x3C (0x5C)	OCR0B	Timer/Counter0 – Output Compare Register B							
0x3B (0x5B)	GIMSK	–	INT0	PCIE1	PCIE0	–	–	–	–
0x3A (0x5A	GIFR	–	INTF0	PCIF1	PCIF0	–	–	–	–
0x39 (0x59)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
0x38 (0x58)	TIFR0		–	–	–	–	OCF0B	OCF0A	TOV0
● ● ●									
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
0x18 (0x38)	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0
0x17 (0x37)	DDRB	–	–	–	–	DDB3	DDB2	DDB1	DDB0
0x16 (0x36)	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0

Making a port to be output

- To make the whole port A output, just do the following:

```
; Making port A to be output
      LDI R16, 0xFF ; All bits in register 16 to one
      OUT 0x1A, R16 ; Write the content of R16 to port register 0x1A
```

- Generate a new project in the simulator, select the ATtiny24 or ATtiny24A and add these three lines behind the Main: line.
- The same does the following:

```
; Making port A to be output
      SER R16 ; All bits in register 16 to one
      OUT DDRA, R16 ; Write the content of R16 to port register DDRA
```

- SER („Set register“) is just an additional mnemonic. The binary produced by the assembler is the same (compare the two binary codes in the listing).
- The use of DDRA as symbol name simplifies the code. The symbol DDRA translates to 0x1A in the assembler. It is defined in the header file tn24def.inc, which is included on top of the source code.

Manipulating single pins

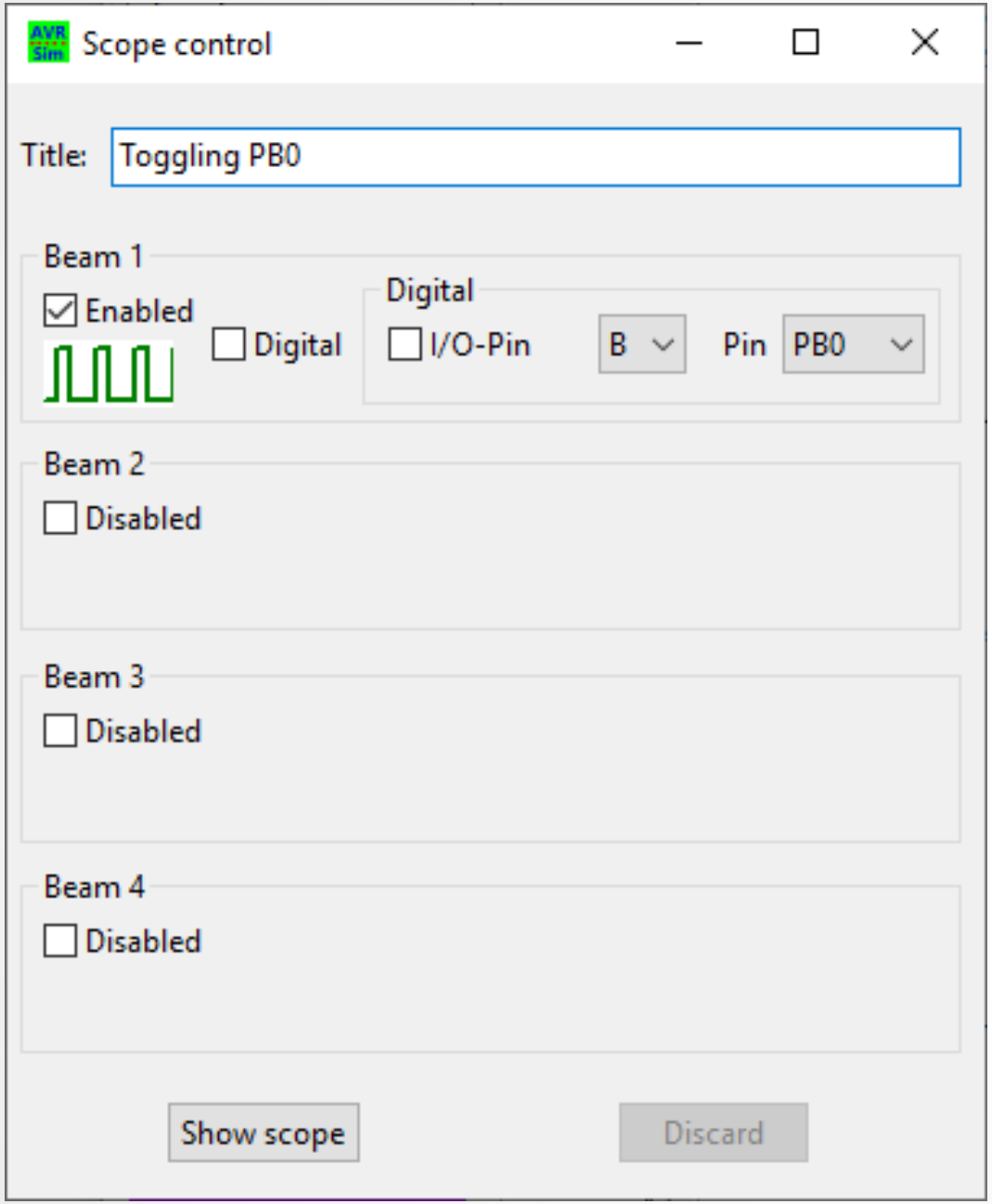
- **If you need to set or clear a single output pin use the following code:**

```
; Making portpin PB0 output and setting it HIGH
      SBI DDRB, DDB0 ; Set the direction bit of PB0 high
      SBI PORTB, PORTB0 ; Set the output port pin PB0 HIGH
```

- **In the simulator, in the „View hardware“ section click the „Ports“ field to view the ports of the controller. In the port window use „Previous“ and „Next“ to switch between port A and port B.**
- **The following switches the port pin PB0 on and off and generates a rectangle on PB0.**

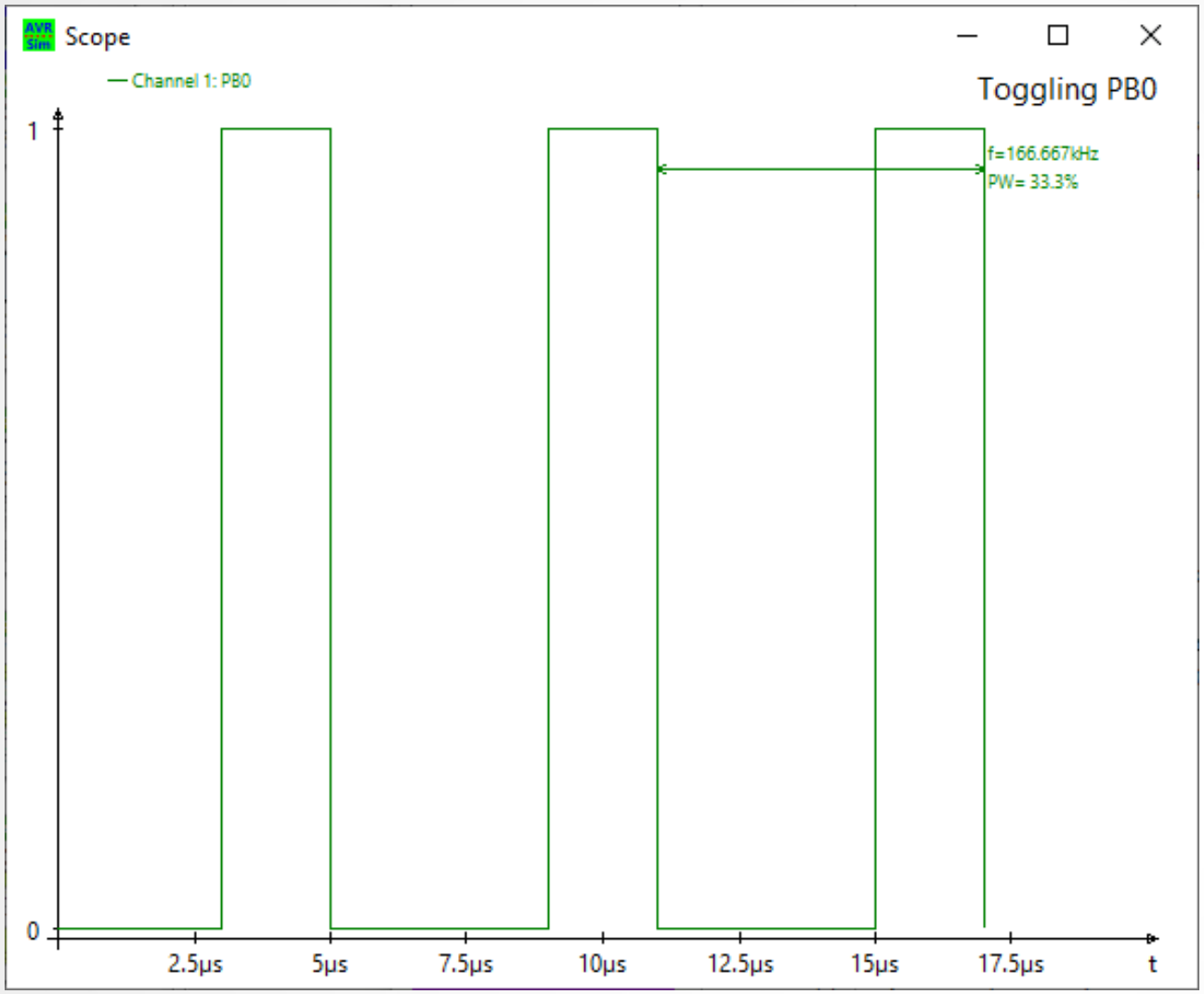
```
; Toggling portpin PB0
      SBI DDRB, DDB0 ; Set the direction bit of PB0 high
Toggle:
      SBI PORTB, PORTB0 ; Set the output port pin PB0 to HIGH
      CBI PORTB, PORTB0 ; Clear the output port pin PB0 to LOW
      RJMP Toggle ; Jump back to the label Toggle
```

Scope view of the toggling pin



- **In the simulation window in the section „View hardware“ enable the field „Scope“.**
- **Click „Disabled“, select „Digital“ and I/O-Pin, „Port B“ and „PB0“. Fill in a title and click „Show scope“.**

Scope display of the toggling



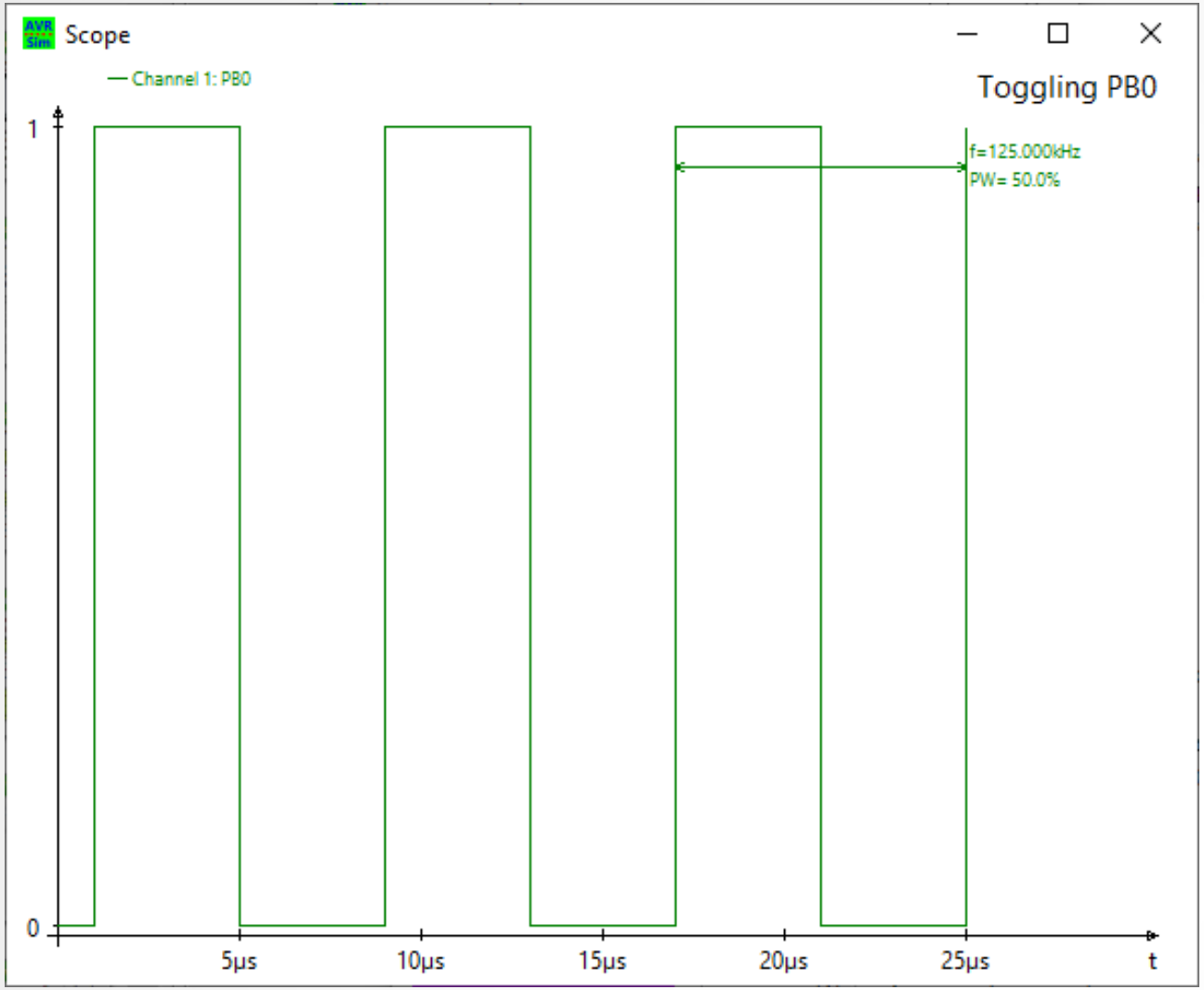
- **If you step through the code, you'll see the rectangle to the left.**
- **The frequency is 167 kHz, the pulse width is 33.3%.**
- **The reason for this is that the set instruction is immediately followed by the clear instruction, while the RJMP is between clear and the next set.**

No operation (NOP)

- The delay introduced by the RJMP has to be compensated to reach a pulse width of 50%.
- An instruction that simply consumes time but does nothing else is NOP. It consumes one clock cycle.
- The RJMP changes the program counter. So the pre-fetch of the controller fails and the RJMP consumes two clock cycles.
- SBI and CBI also consume two clock cycles each because the content of DDRB has to be read and the DDB0 bit has to be set and written back to DDRB.
- To compensate the RJMP we add two NOP instructions.

```
; Toggling portpin PB0 with 50% pulse width
      SBI DDRB, DDB0 ; Set the direction bit of PB0 high
Toggle:
      SBI PORTB, PORTB0 ; Consumes two clock cycles
      NOP ; Consumes one clock cycle
      NOP ; Consumes one clock cycle
      CBI PORTB, PORTB0 ; Consumes two clock cycles
      RJMP Toggle ; Consumes two clock cycles
```

The ideal rectangle



- **Now the rectangle has exactly 50% pulse width.**
- **Because the whole execution cycle is eight clock cycles long and as the ATtiny24 is clocked with 1 MHz by default, the frequency is 1 MHz/8 = 125 kHz.**

Prolonging the cycle

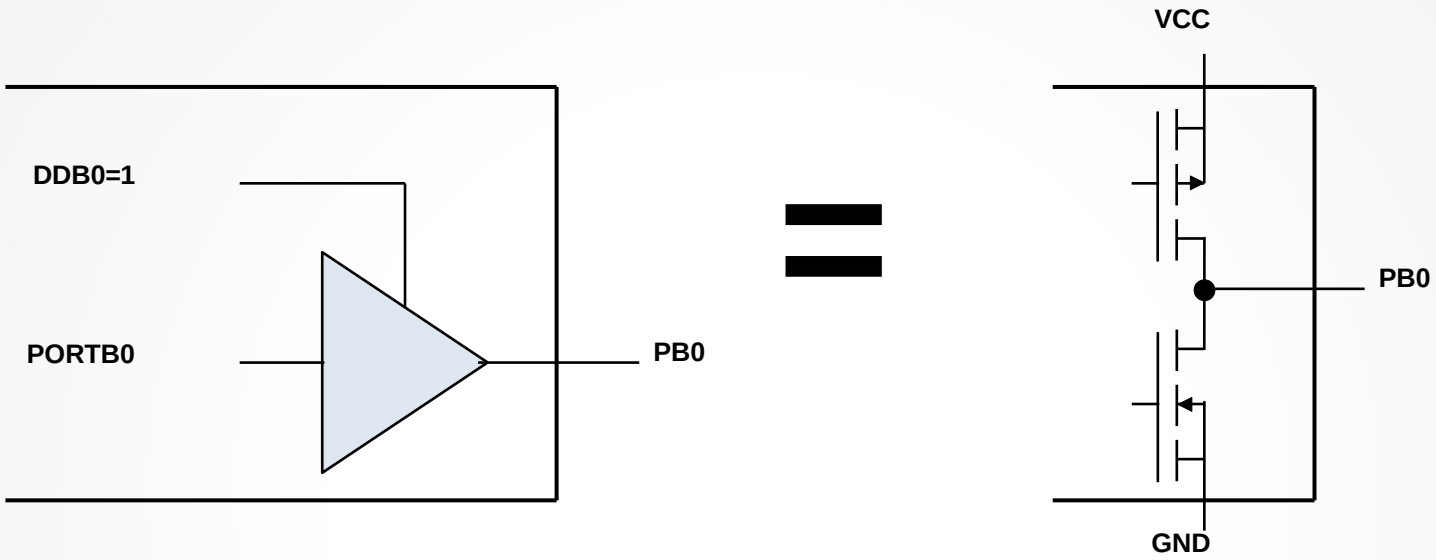
- **We can add further NOPs to prolong the cycle.**

```
; Toggling portpin PB0 at 100 kHz with 50% pulse width
      SBI DDRB, DDB0 ; Set the direction bit of PB0 high
Toggle:
      SBI PORTB, PORTB0 ; Consumes two clock cycles
      NOP ; Consumes one clock cycle
      NOP ; Consumes one clock cycle
      NOP ; Add another clock cycle during HIGH
      CBI PORTB, PORTB0 ; Consumes two clock cycles
      NOP ; And another clock cycle during LOW
      RJMP Toggle ; Consumes two clock cycles
```

- **As now the whole cycle is 10 clock cycles long, the frequency is 1 MHz / 10 = 100 kHz.**
- **We can add further NOPs to prolong the cycle. But the effect is rather limited and we cannot reach really small frequencies.**
- **Very low frequencies require longer delay times.**

Properties of output pins

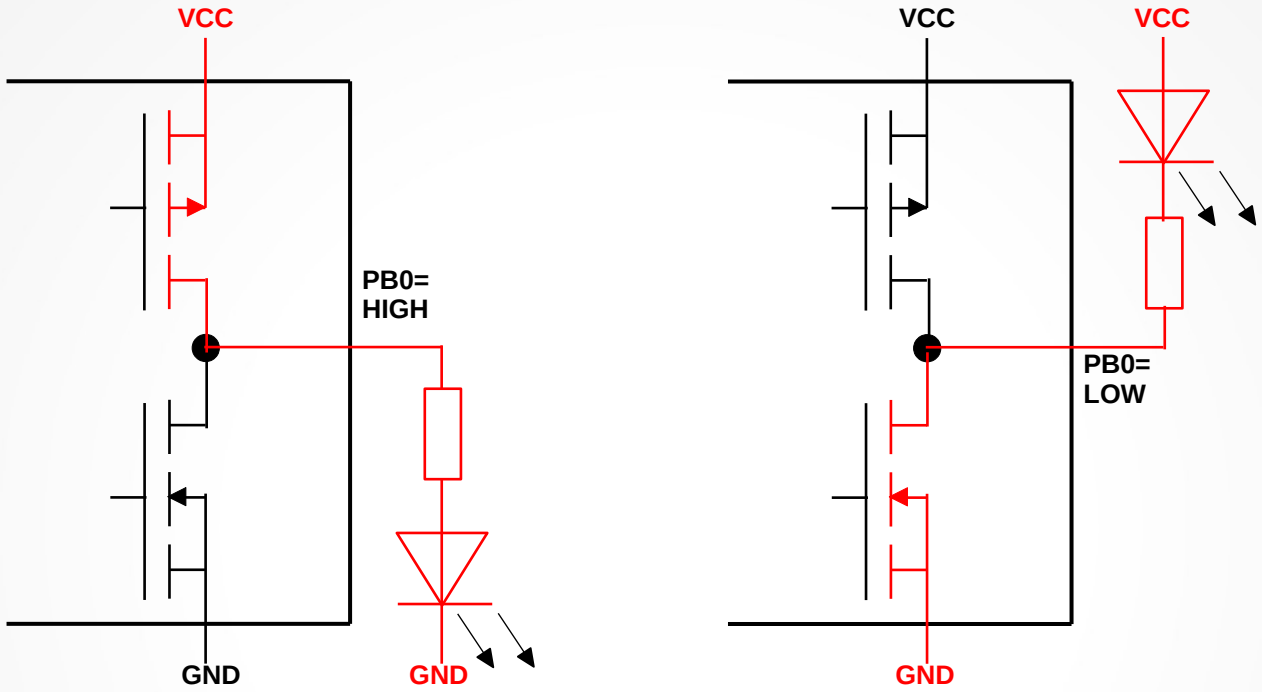
- **The output pins are designed as follows:**



- **The DD-bit switches the output driver on/off.**
- **If the DD-bit is on, the output follows the state of the PORT bit.**
- **The output pins are driven by two transistors.**

Outputs driving High and Low

- **The driving transistors can provide current in both directions:**



- **If the output pin is high, the upper transistor connects the output pin to VCC. An external component can source current to GND.**
- **If the output pin is low, the lower transistor connects the output pin to GND. An external component can sink current from VCC through the output pin.**

Further properties of outputs

- **The outputs are protected against short circuits.**
- **They can sink or source up to 40 mA.**
- **Voltage losses are between 0.5 to 0.7 Volt at 10 mA.**

Pins as inputs

- If the DD bit is low, the pin serves as input.
- If the respective port bit is high, a pull-up resistor of around 50 kΩ forces the input pin high, unless externally pulled low.
- The current state of the pin can be read in the PIN port register.

```
; Reading port PINB
      IN R16, PINB ; Reading all input pins in port B
; Reading portpin PB0 and jumping over the next instruction if HIGH
      SBIS PINB,PINB0 ; Read PINB0 and jump if set (HIGH)
      RJMP PinIsLow
; Reading portpin PB0 and jumping over the next instruction if LOW
      SBIC PINB,PINB0 ; Read PINB0 and jump if clear (LOW)
      RJMP PinIsHigh
```

- In most of the AVRs writing a ONE to the PIN bit toggles the respective PORT bit.

Conclusions

- **Internal hardware can be configured by setting or clearing a few bits in port registers.**
- **All properties of the internal hardware can be changed at any time.**
- **Pins of AVRs can be used as output as well as inputs.**
- **The state of input pins can be used to branch and execute depending program parts.**

Questions and Tasks for Lecture 2

Question 2-1: In the register summary are four portpins listed for PORTB. Where is the fourth portpin? Why is it not displayed in the schematic?

(Hints: Use the Device‘s Data Book for the complete schematic and to find out why PB3 is not listed here.)

Questions and Tasks for Lecture 2, Continued

Question 2-2: Here the ATtiny24 is shown and discussed. There is another controller named ATtiny24A. What are the differences between those two types in respect to

a) Memory sizes (Flash, SRAM, EEPROM),

b) Operating voltages,

c) Clock frequencies,

d) Electrical characteristics.

(Hint: Use the Data Books for these two types provided by Microchip)

Questions and Tasks for Lecture 2, Continued

Question 2-3: There are LEDs available that can light in two different colors (e.g. red and green), but they only have two pins. What can be done to attach such a LED

a) to two output pins, or

b) to a single output pin

of an AVR at 5 Volt operating voltage?

(Hint for b: The forward voltage of LEDs is approximately 2 Volt. Also consider the voltage drops when current is sourced/sinked!)

Bonus question: What would be necessary to have the LED lighting with 20% intensity in red and 80% intensity in green?