

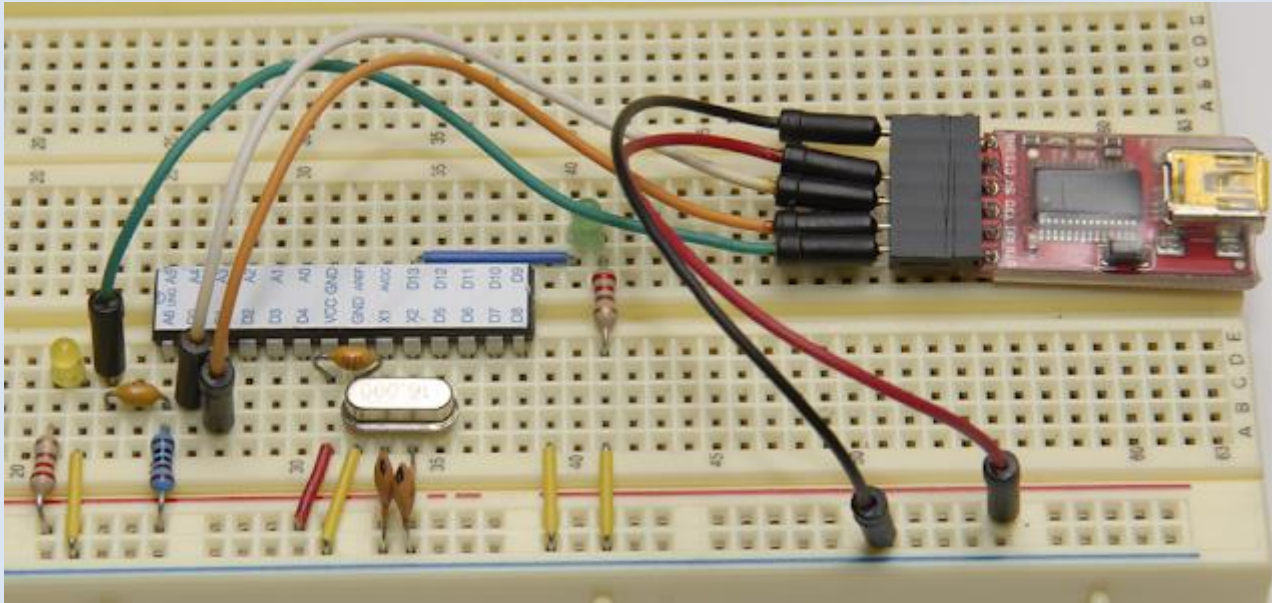
Mikroprozessor ATmega328

- Arduino Standalone
- Arduino IDE Erfahrungen
- Rechteckschwingung Arduino (*.ino *.cpp)
- ATmega328P Daten, Externe Anschlüsse
- AVR Architektur
- Memory, SRAM Data Memory
- Arithmetic logic unit, Signalprocessing
- Register PORTB
- I/O-Port programmieren
- Mnemonic SBI, OUT
- GNU AVR-GCC
- Assembler: Quelle zum Opcode
- Gerd's AVR Simulator
- Programm schreiben, Assembler-Code
- Assembler-Code-Listing
- Simulation SIM_02
- Scope SIM_02, SIM_05
- Cycles
- Informationen

Thema

ATmega328p	Den ATmega328p Microcontroller (μ C) in seinen Funktionen kennen lernen.
Idee	AVR-Simulation
Aufgabe	Maximale Frequenz einer Rechteckschwingung? Experimentierbord mit ATmega328p Standalone. Erfahrungen mit Arduino. Physikalische Pins des ATmega328p. Interner Aufbau des ATmega328. Organisation der Speicher. CPU und Register Sprache Assembler Gerd's AVR Simulator Rechteckschwingung, Takte
Danksagung	Das Material von Gerhard Schmidt, DG4FAC, war sehr hilfreich bei der Erarbeitung des Themas. http://www.dg4fac.de/ http://www.avr-asm-tutorial.net/

Arduino Standalone



FTDI Tx an IC Pin #2 (Tx an Rx)
FTDI Rx an IC Pin #3 (Rx an Tx)
FTDI DTR an Kondensator
0,1 µF und weiter an IC Pin 1
FTDI Gnd an Steckbrett
FTDI 5V an Steckbrett

Quelle:

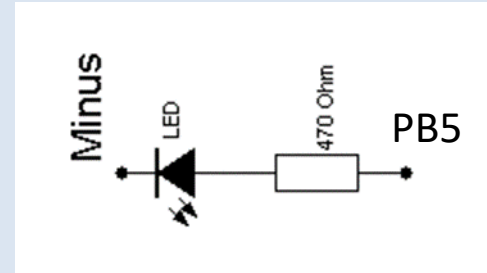
<https://www.yuriystoys.com/2012/02/arduino-on-beadboard-uploading-your.html>

Arduino IDE Erfahrungen

Port und Pins

Arduino Pin:
Mikrokontroller:
(physical pin)
Port
Port Pin:
Clock frequency

Pin 13
Pin 19
B
PB5 (PORTB5)
16 MHz



Arduino IDE Sketch

```
void setup() {  
  pinMode(13, OUTPUT); // Arduino PIN 13  
}  
  
void loop() {  
  digitalWrite(13, HIGH);  
  delay(1000);  
  digitalWrite(13, LOW);  
  delay(1000);  
}
```

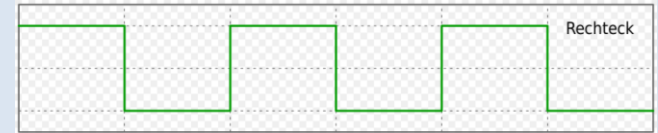
Wie aus C++ Code ein
maschineller Code
wird, bleibt verborgen!

Flash

924 Bytes

Rechteckschwingung Arduino (*.ino)

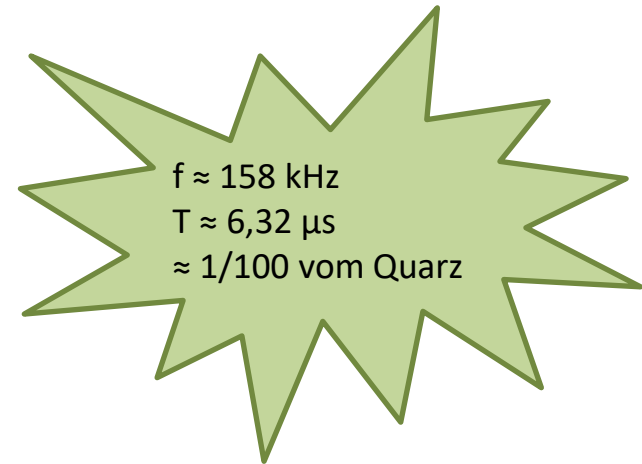
Aufgabe:	Rechteckschwingung	Ziel maximale Frequenz?
Port und Pins:	Arduino Pin:	Pin 13
Mikrokontroller:	(physical pin)	Pin 19
	Port	B
	Port Pin:	PORTB5 (PB5)
	Port Direction Pin	DDB5
	Clock frequency	16 MHz



Arduino IDE Sketch

```
boolean status = 0;
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  status = status ^ 1;
  digitalWrite(13, status);
}
```

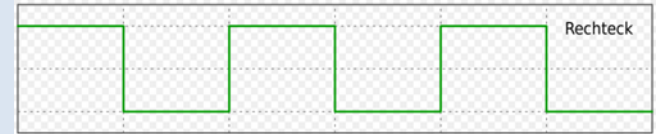


Flash

750 Bytes

Rechteckschwingung Arduino (*.cpp)

Aufgabe:	Rechteckschwingung	Ziel maximale Frequenz?
Port und Pins:	Arduino Pin:	Pin 13
Mikrokontroller:	(physical pin)	Pin 19
	Port	B
	Port Pin:	PORTB5 (PB5)
	Port Direction Pin	DDB5
	Clock frequency	16 MHz

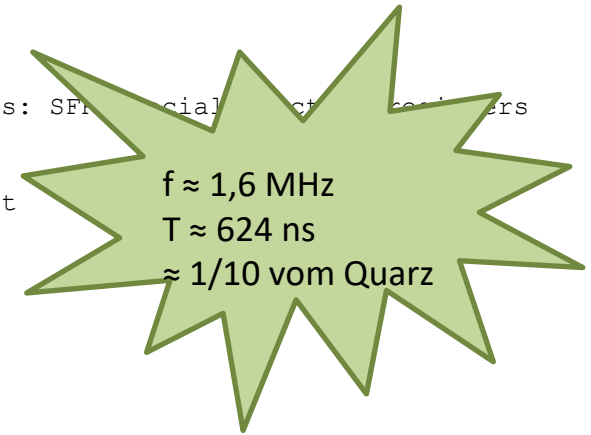


Arduino IDE Sketch

```
// Die *.ino-Datei bleibt bestehen, nur Inhalte löschen
// Zweiter Tab mit main.cpp

#include <avr/io.h> // Defines names for AVR registers: SFR and special function registers

int main(void) {
    DDRB |= (1 << DDB5); // make PORTB5 as output
    for(;;) {
        PORTB ^= (1 << PORTB5); // toggle Pin PORTB5
    }
    return 0; // the program executed successfully
}
```



Flash

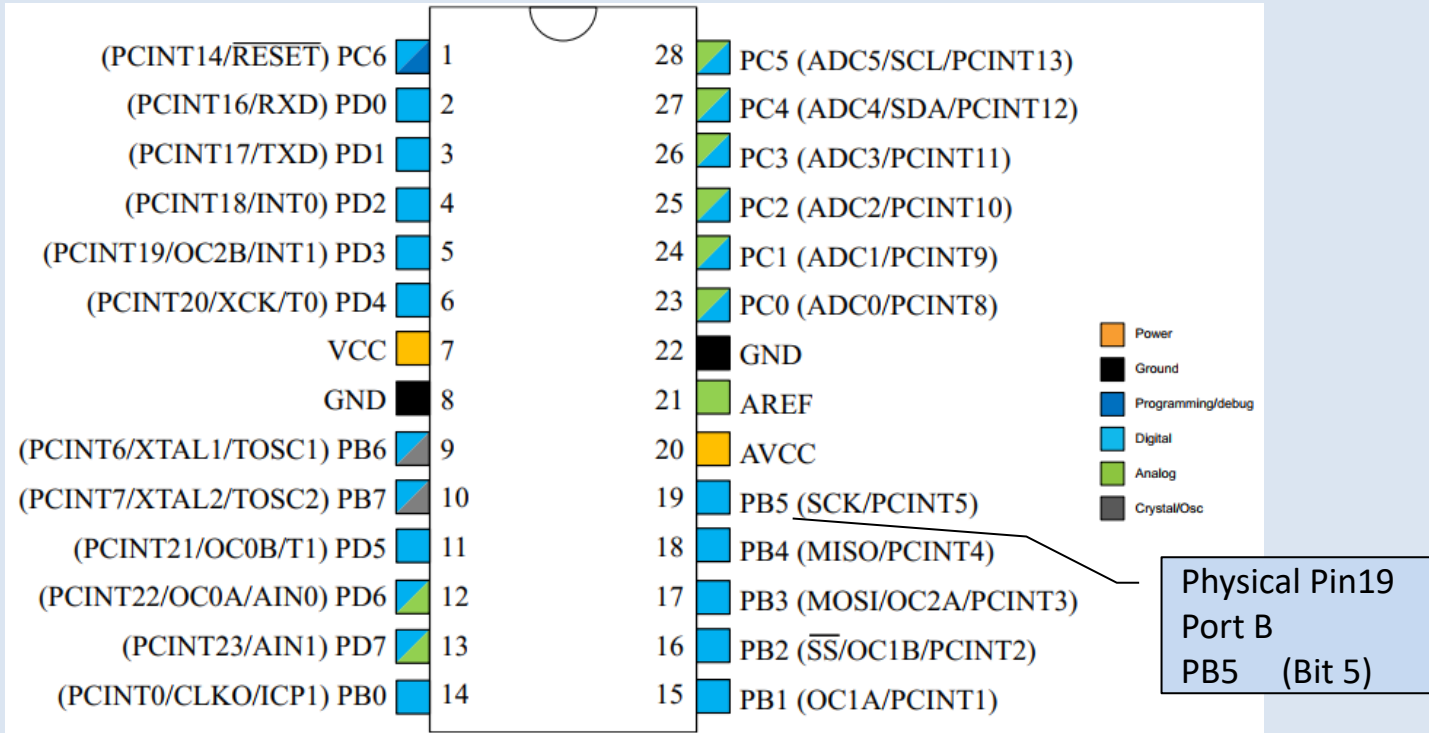
144 Bytes

ATmega328P Daten

Hersteller	Microchip (vormals Atmel)
Familie	8-Bit-Mikrocontroller
Architektur	RISC
Anzahl Instruktionen	131
Register	32 x 8
Quarz	16 MHz
	16 MIPS at 16 MHz
Flash Programm Speicher	32 KiB kibibyte
EEPROM	1 KiB kibibyte
SRAM	2 KiB kibibyte
Peripherie	Timer/Counter; PWM; ADC; USART; SPI; I ² C
Betriebsspannung	2,7 V bis 5,5 V

Quelle: © 2018 Microchip Technology Inc. Data Sheet Complete DS40002061A-page 1

Externe Anschlüsse



Quelle: Atmel-42735A-ATmega328/P_Datasheet_Complete-06/2016

AVR Architektur

Harvard-Architektur
RISC

Program **Flash** Memory for Application ...

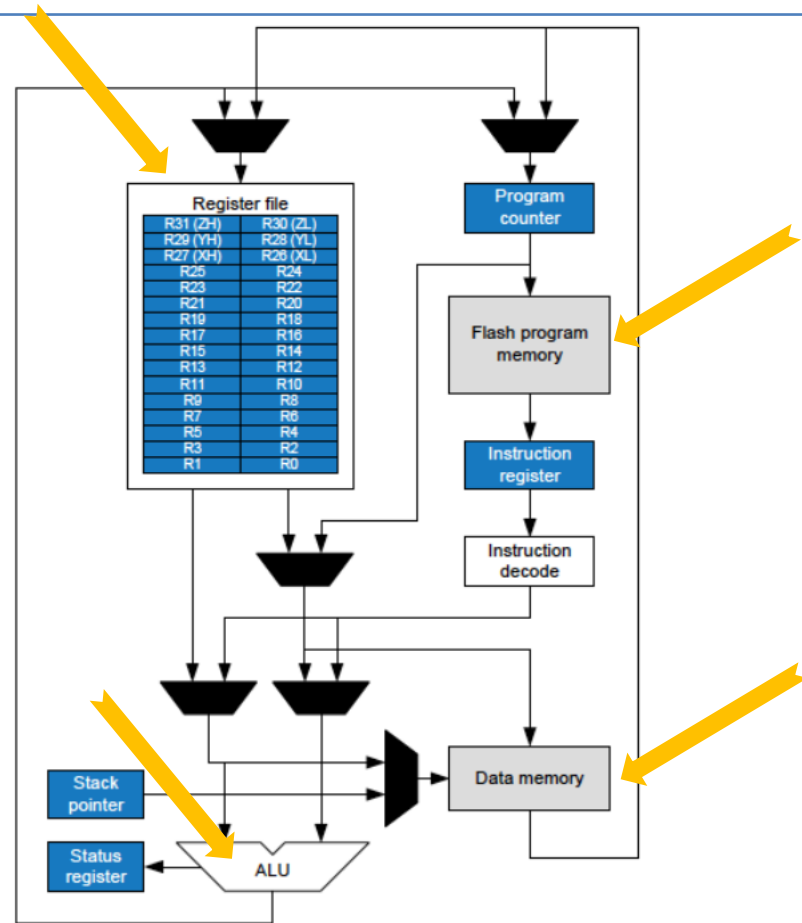
Data Memory

- 32 x 8 General Purpose Working Registers
- I/O memory space with 64 addresses, **Ports**
- Internal **SRAM**

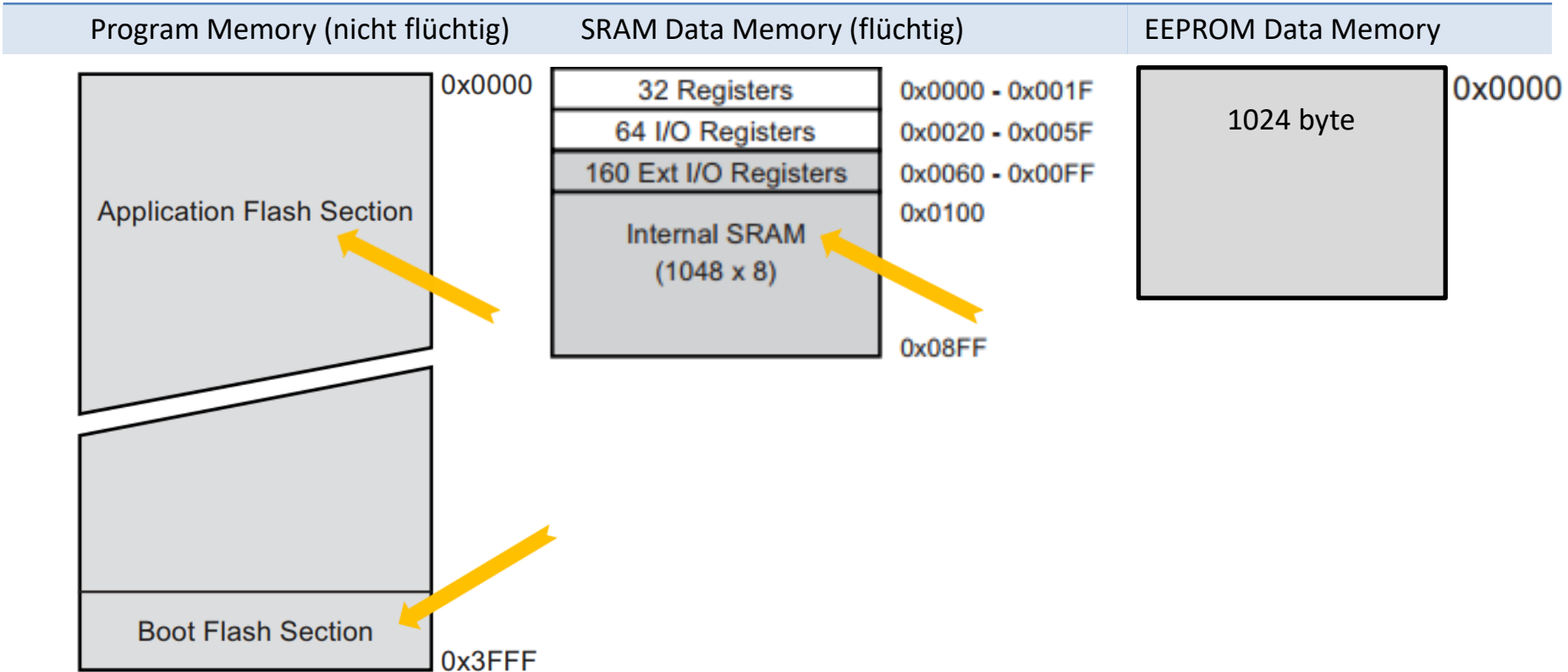
ALU supports arithmetic and logic operations between registers or between a constant and a register

Digital Signalprocessing

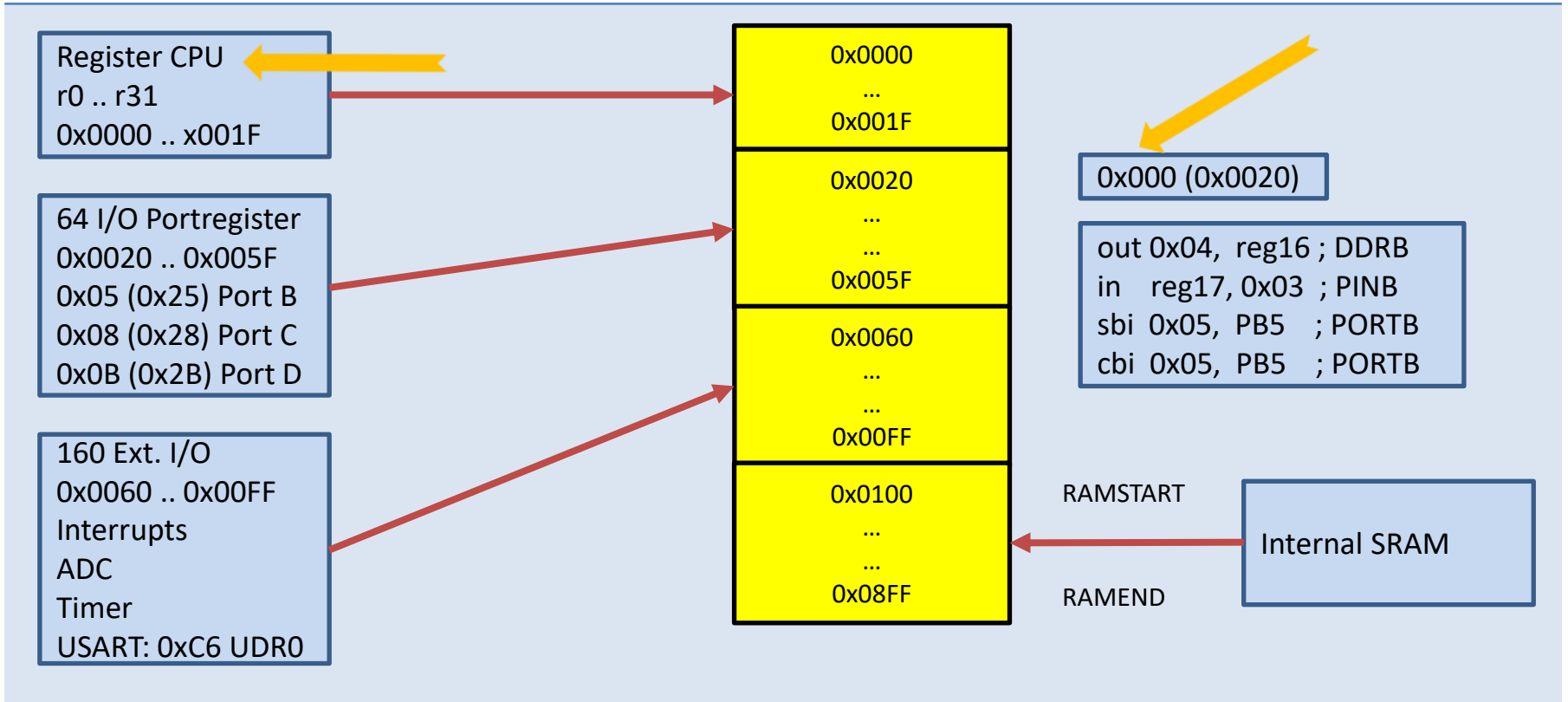
Quelle: <https://microchipdeveloper.com/8avr:avrcore>



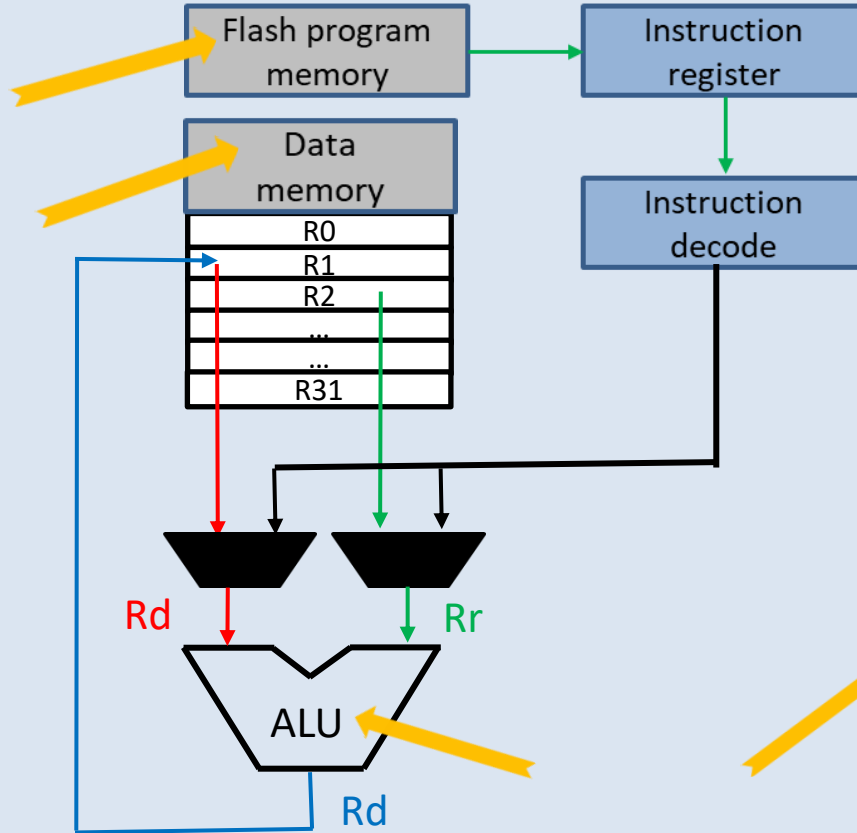
Memory



SRAM Data Memory



Arithmetic logic unit



Innerhalb eines Taktes werden arithmetische Operationen zwischen General Purpose Registern (oder einem Register und einer Konstanten) ausgeführt.

Die Operationen haben die Kategorien: Arithmetisch, logisch, bit-Operationen

Beispiel:

add r1,r2 ; Add r2 to r1 ($r1=r1+r2$)

ADD Rd,Rr

$Rd \leftarrow Rd + Rr$

$0 \leq d \leq 31, 0 \leq r \leq 31$

Signalprocessing

Beispiel:

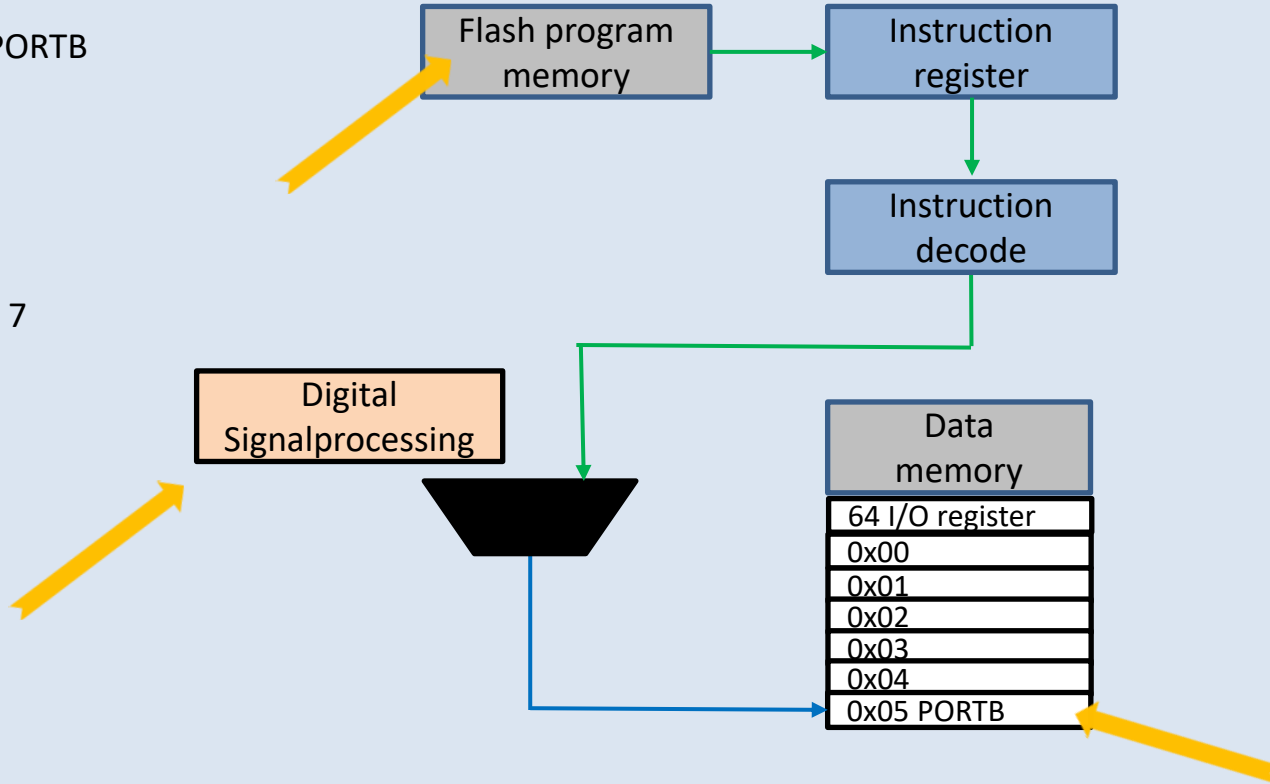
Setze High Bit in PORTB
`sbi 0x05, 0x01`

Syntax:

`SBI A,b`

$I/O(A,b) \leftarrow 1$

$0 \leq A \leq 31, 0 \leq b \leq 7$



Register PORTB

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
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	

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
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	

PINB – The Port B Input Pins Address

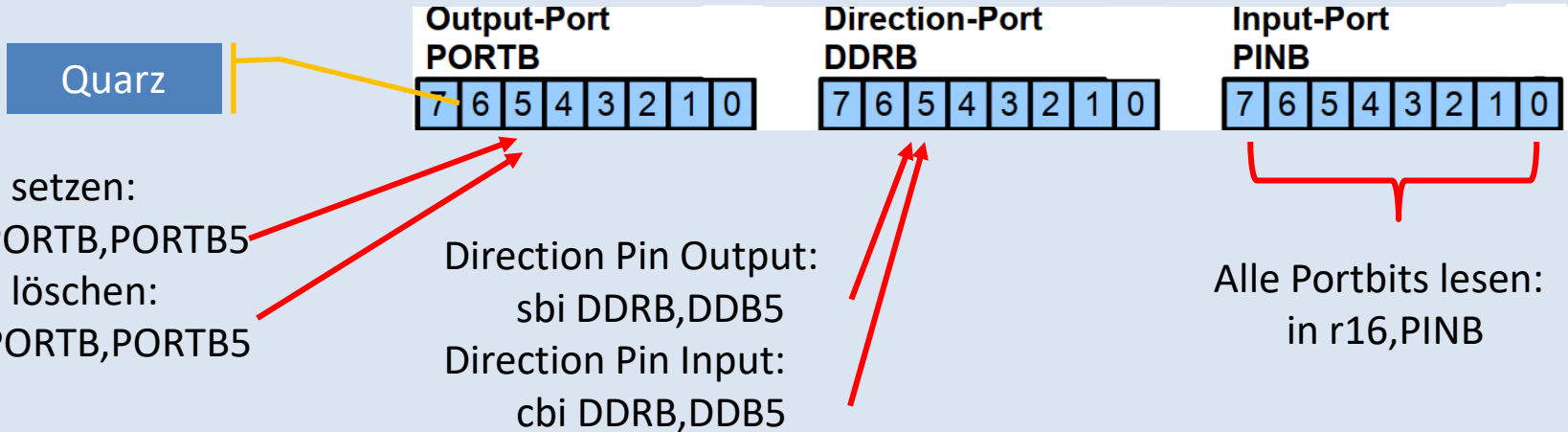
Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Quelle: © 2018 Microchip Technology Inc. Data Sheet Complete DS40002061A-page 100

I/O-Port programmieren

ATmega328P Ports und Pins:
PORTB: PORTB0 bis PORTB5
PORTC: PORTC0 bis PORTC5
PORTD: PORTD0 bis PORTD7

I/O-Pin
Arduino Pin 13
Mikrokontroller Pin 19
Port Pin PORTB5



Mnemonic SBI

Description

Sets a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

Operation:

(i) $I/O(A,b) \leftarrow 1$

Syntax:

(i) SBI A,b

Operands:

$0 \leq A \leq 31, 0 \leq b \leq 7$

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

1001	1010	AAAA	Abbb
------	------	------	------

Example:

```
sbi 0x1C,0 ; Set read bit in EECR
```

Words: 1 (2 bytes)

Cycles: 2

Mnemonic OUT

Description

Stores data from register Rr in the Register File to I/O space.

Operation:

(i) $I/O(A) \leftarrow Rr$

Syntax:

(i) OUT A,Rr

Operands:

$0 \leq r \leq 31, 0 \leq A \leq 63$

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

1011	1AAr	rrrr	AAAA
------	------	------	------

Example:

```
clr r16 ; Clear r16
ser r17 ; Set r17
out 0x18,r16 ; Write zeros to Port B
nop ; Wait (do nothing)
```

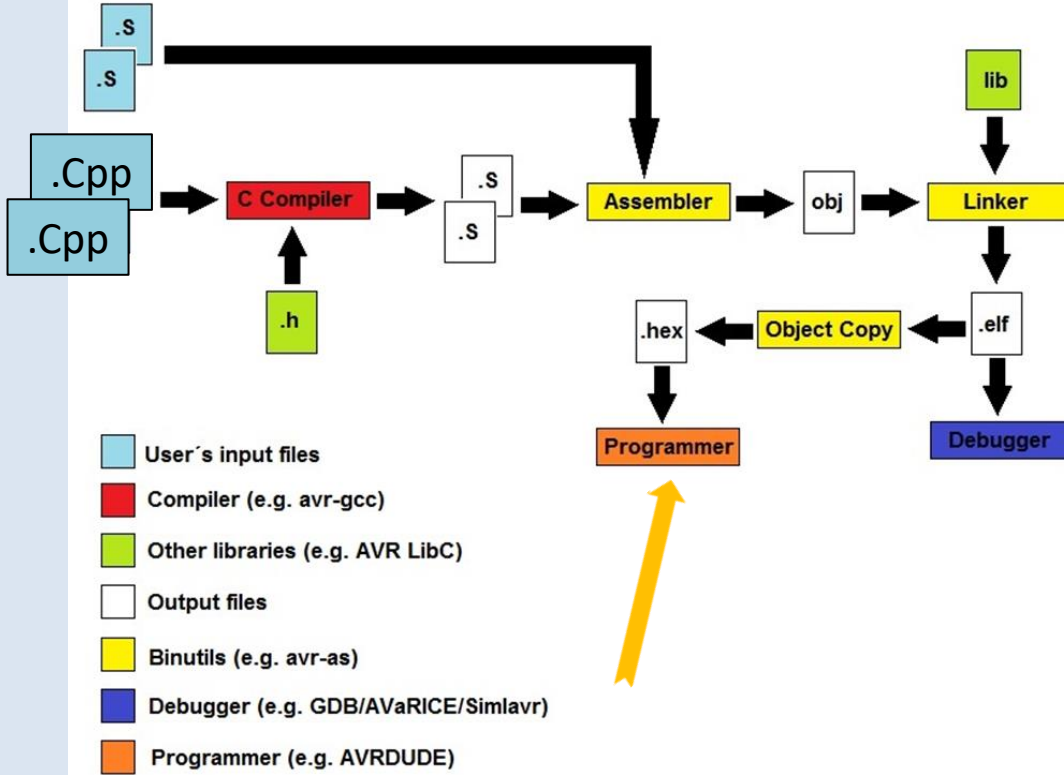
Words: 1 (2 bytes)

Cycles: 1

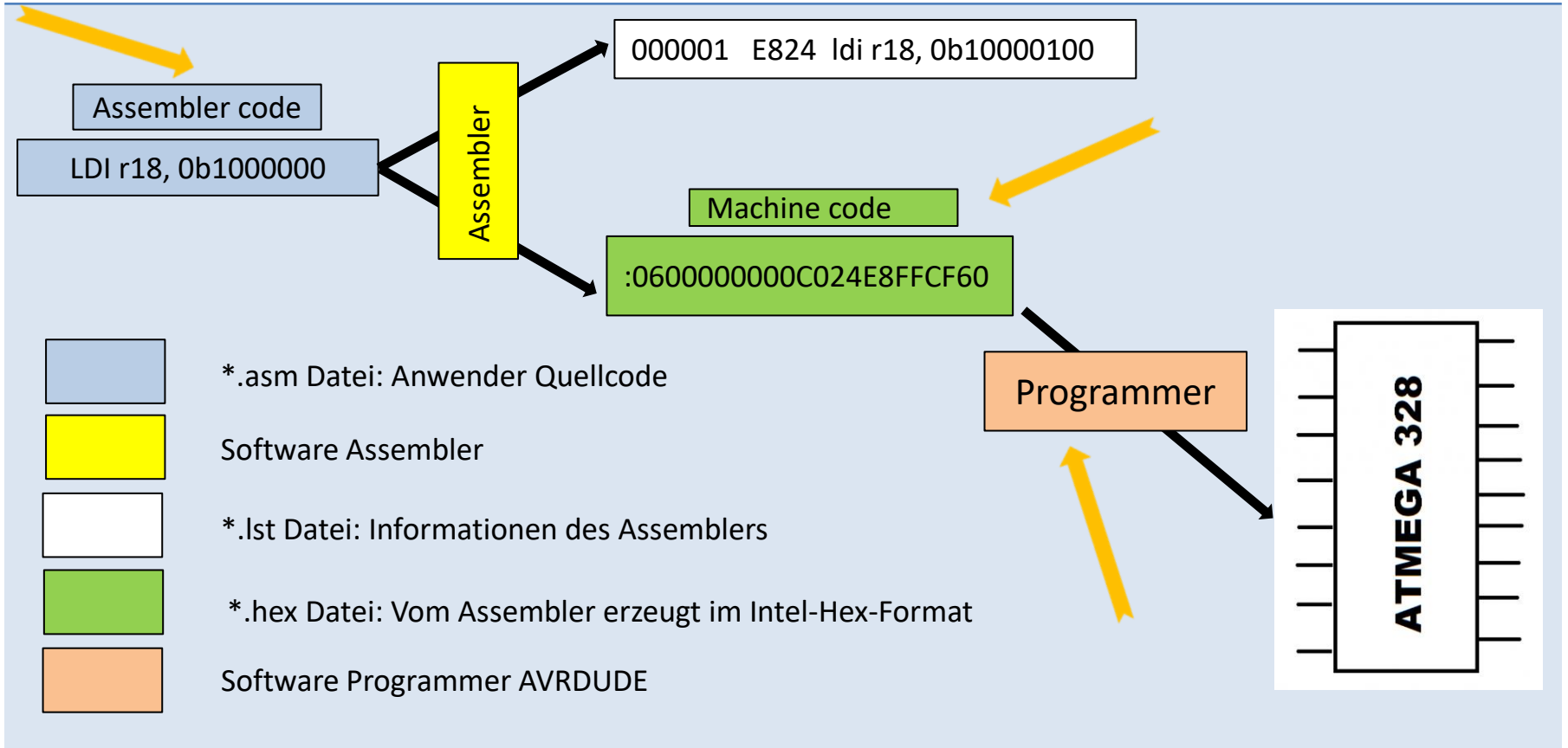
GNU AVR-GCC

Arduino-IDE

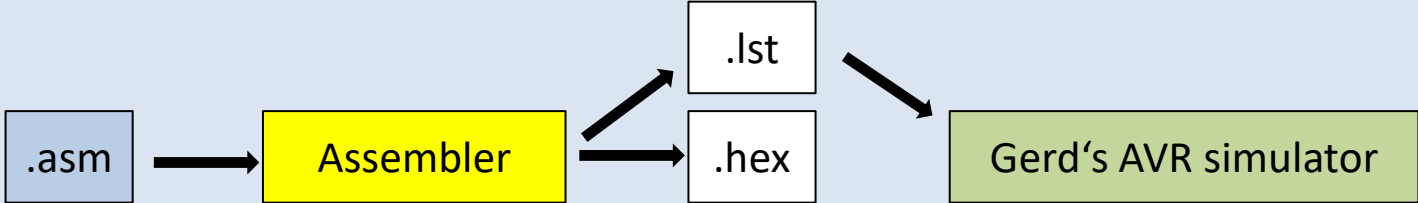
.ino



Assembler: Quelle zum Opcode



Gerd's AVR Simulator



Anwender Eingabe Datei



gavrasm Assembler



Ausgabe Datei Intel-Hex-Format



Gerd's AVR Simulator

Programm schreiben

Gerd's AVR Simulator	Starten (Zur detaillierte Anwendung siehe u. g. Handbuch.)
----------------------	---

	Projekt anlegen
--	-----------------

	Assembler-Programm in Editor eingeben
--	---------------------------------------

	Quellcode assemblieren
--	------------------------

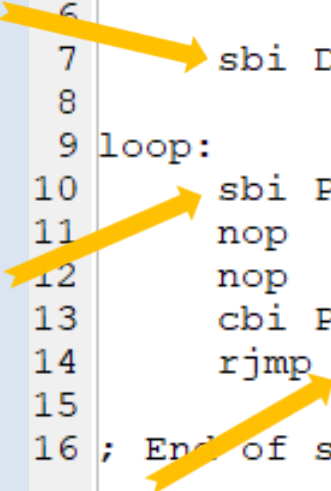
	Simulation
--	------------

Quelle:	http://www.avr-asm-tutorial.net/avr_sim/index_en.html http://www.avr-asm-tutorial.net/avr_sim/23/avr_sim_Handbuch_v23.pdf
---------	--

Assembler-Code

SIM_02.asm

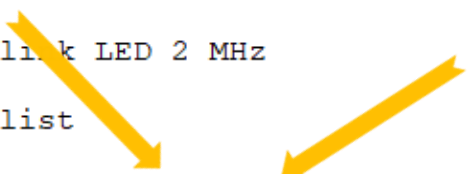
```
1 ; Blink LED 2 MHz
2
3 .nolist
4 .include "m328pdef.inc" ; get definitions of ATmega328P
5 .list
6
7     sbi DDRB, PB5        ; set bit PB5 in Direction-Port B
8
9 loop:                               ; jump target label "loop"
10    sbi PORTB, PB5       ; set bit PB5 in Output-Port B
11    nop                  ; do nothing
12    nop                  ; do nothing
13    cbi PORTB, PB5       ; clear bit PB5 in Output-Port B
14    rjmp loop            ; relative jump to label "loop"
15
16 ; End of source code
```



Assembler-Code-Listing

SIM_02.lst

```
1 gavrasm Gerd's AVR assembler version 4.8 (C)2020 by DG4FAC
2 -----
3
4 Path:          C:\Users\enno_\Desktop\Assembler-EBW\GerdsSimulator\SIM_02\
5 Source file:   SIM_02.asm
6 Hex file:      SIM_02.hex
7 Eeprom file:   SIM_02.eep
8 Compiled:      09.05.2021, 12:38:34
9 Pass:         2
10
11      1: ; Blink LED 2 MHz
12      2:
13      3: .nolist
14      6:
15      7: 000000    9A25  sbi DDRB, PB5      ; set bit PB5 in Direction-Port B
16      8:
17      9: loop:                                ; jump target label "loop"
18     10: 000001    9A2D  sbi PORTB, PB5    ; set bit PB5 in Output-Port B
19     11: 000002    0000  nop                ; do nothing
20     12: 000003    0000  nop                ; do nothing
21     13: 000004    982D  cbi PORTB, PB5    ; clear bit PB5 in Output-Port B
22     14: 000005    CFFB  rjmp loop         ; relative jump to label "loop"
23     15:
24     16: ; End of source code
```



Simulation SIM_02

```
15      7: 000000   9A25  sbi DDRB, PB5      ; set bit PB5 in Direction-Port B
16      8:
17      9: loop:
18      10: 000001   9A2D  sbi PORTB, PB5       ; set bit PB5 in Output-Port B
```

avr_sim SIM_02

Restart Step Skip Run/Go Stop

Simulation status

Prog counter = \$000001
Instructions = 1
Stackpointer = \$0000
Watchdog = 0.000000%
Clock frequ. = 16,000,000
Time elapsed = 125.0 ns
Stop watch = 125.0 ns
Sleep share = 0.000000%

SREG

I	T	H	S	V	N	Z	C
0	0	0	0	0	0	0	0

Update status Instructions: 1000 Step Delay ms: 10

Register

Reg	+0	+1	+2	+3	+4	+5	+6	+7
R0	00	00	00	00	00	00	00	00
R8	00	00	00	00	00	00	00	00
R16	00	00	00	00	00	00	00	00
R24	00	00	00	00	00	00	00	00

Messages

\$0000: Starting

Show internal hardware

Ports Timers WDT ADC EEPROM SRAM

Tools

Scope Alert

Port

(Help)	7	6	5	4	3	2	1	0
PORTB	0	0	0	0	0	0	0	0
DDRB	0	0	1	0	0	0	0	0
PINB	0	0	Lo	0	0	0	0	0
TINnB								
INTnB								
PCINT0	7	6	5	4	3	2	1	0

Previous **B** Next

Scope SIM_02

Ergebnis

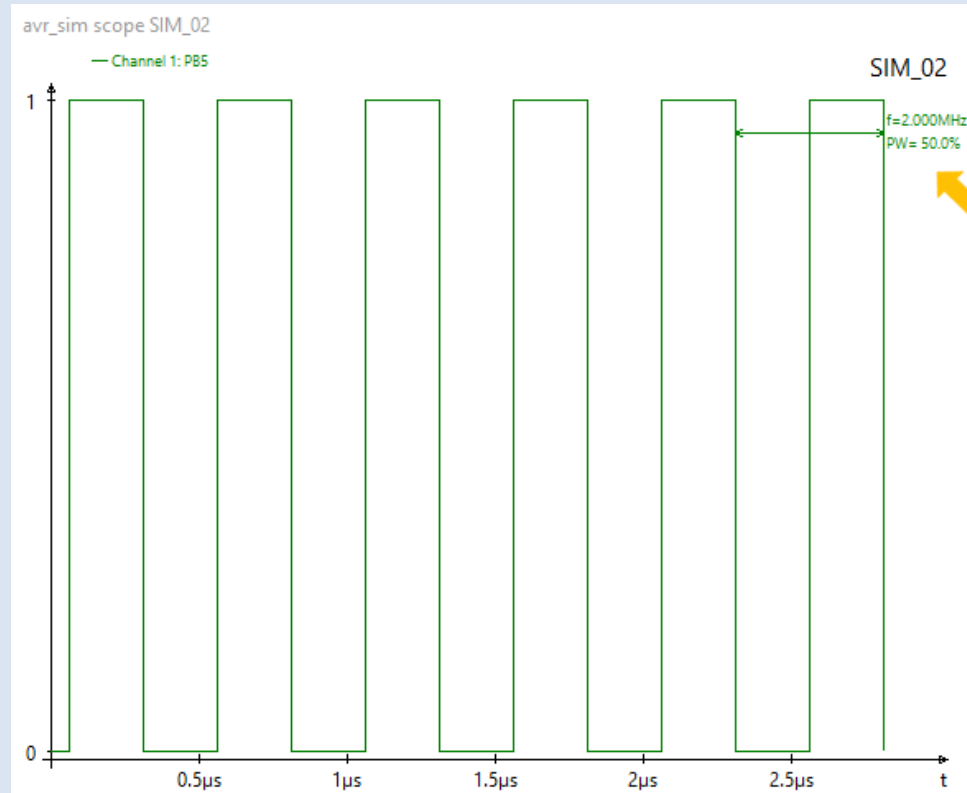
$f = 2 \text{ MHz}$

$T = 500 \text{ ns}$

2 MHz / 16 MHz

1 / 8

8 Takte (cycles)



Scope SIM_05

Ergebnis

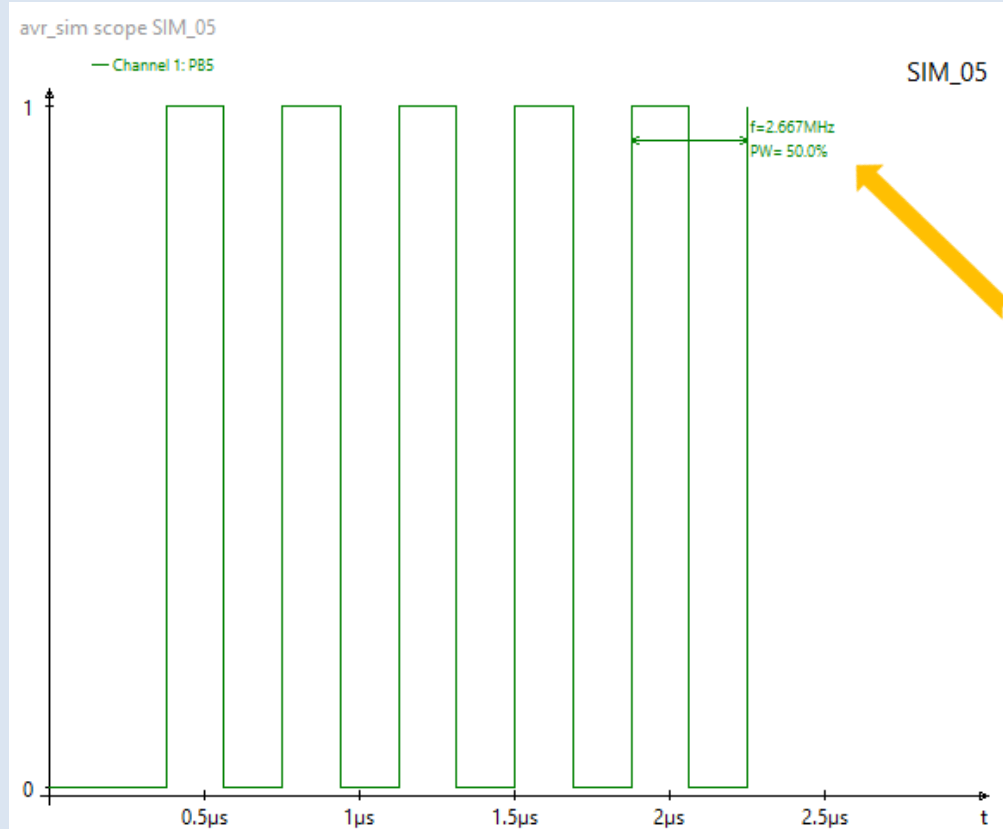
$f = 2,667 \text{ MHz}$

$T = 375 \text{ ns}$

$2,667 \text{ MHz} / 16 \text{ MHz}$

$1 / 6$

6 Takte (cycles)



Cycles

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2					f	16.000.000	1/s									
3					T=1/f	0,0000000625	s									
4						0,0000625000	ms		0,0625	us		62,5	ns			
5	time at start of loop					0,0001250000	ms		0,125	us		125	ns			
6																
7																0 ns
8	SBI DDRB		2		cycle	0,0001250000	ms		0,125	us		125	ns		125 ns	
9																
10	SBI PORTB		2		cycle	0,0001250000	ms		0,125	us		125	ns		250 ns	
11	NOP		1		cycle	0,0000625000	ms		0,0625	us		62,5	ns		312,5 ns	
12	NOP		1	4	cycle	0,0000625000	ms		0,0625	us		62,5	ns		375 ns	
13	CBI PORTB		2		cycle	0,0001250000	ms		0,125	us		125	ns		500 ns	
14	RJMP		2	4	cycle	0,0001250000	ms		0,125	us		125	ns		625 ns	
15																
16	loop											500	ns			
17					f=1/T							2.000.000,00	Hz			
18												2.000,00	kHz			
19												2,00	MHz			
20																

Hardware

Arduino UNO

Arduino UNO Board mit ATmega328p.

Standalone

ATmega328p auf Steckbrett

USB zu TTL Adapter

“FTDI232 USB to TTL Serial Adapter Module”
zur Programmierung über die serielle Schnittstelle

ISP

ISP (In-System-Programmer)

Arduino auf Breadboard

Idee	Standalone ATmega328p
Arduino Standalone	https://www.arduino.cc/en/Main/Standalone https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard https://www.mikrocontroller.net/articles/AVR-Tutorial: Equipment
FTDI Adapter	https://www.yuriystoys.com/2012/02/arduino-on-beadboard-uploading-your.html
Arduino ISP	https://www.arduino.cc/en/Main.ArduinoISP https://www.arduino.cc/en/uploads/Main/ArduinoISP_WindowsDrivers.zip

Assembler Compiler

Simulator

Gerd's AVR assembler

http://www.avr-asm-tutorial.net/avr_sim/index_en.html

http://www.avr-asm-tutorial.net/avr_sim/23/avr_sim_23_win64_debug.zip

Programmiersoftware:
AVRDUDE GUI

<https://blog.zakkemble.net/avrdudess-a-gui-for-avrdude/>

Programmiersoftware:
AVRDUDE 6.3

Vorhanden in:

...\arduino-1.8.12\hardware\tools\avr\bin\avrdude.exe

Konfigurationsdatei in:

...\arduino-1.8.12\hardware\tools\avr\etc\avrdude.conf

Tutorials

AVR-GCC-Tutorial	https://www.mikrocontroller.net/articles/AVR-GCC-Tutorial
WinAVR	https://www.mikrocontroller.net/articles/WinAVR
ArduinoISP	https://www.arduino.cc/en/Guide/ArduinoISP
AVR-Assembler lernen	http://www.avr-asm-tutorial.net/avr_de/absolute_beginner/starten/starten.html
Unterrichtsmaterial	https://www.rahner-edu.de/grundlagen/avr-assembler-teil-0/
AVR Assembler Einführung	https://rn-wissen.de/wiki/index.php?title=AVR Assembler Einf%C3%BChrung
Anfängerkurs G. Schmidt	http://www.avr-asm-download.de/beginner_de.pdf
AVR beginners	http://avrbeginners.net/
MICROCHIP	https://onlinedocs.microchip.com/pr/GUID-E06F3258-483F-4A7B-B1F8-69933E029363-en-US-2/index.html
MICROCHIP Defeloper	https://microchipdeveloper.com/8avr:start
Block Diagram	https://www.avrfreaks.net/forum/atmega328p-alu-and-multiplier-question?page=all