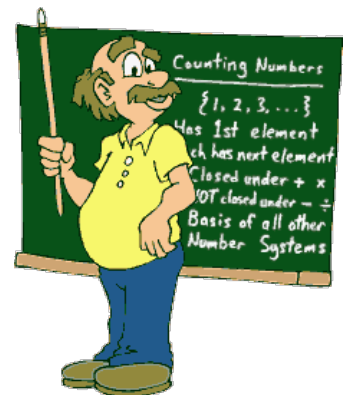


IECA

Embedded Computer Architecture

Lesson 6: Status register and delays



Unconditional "long" Jump (JMP)

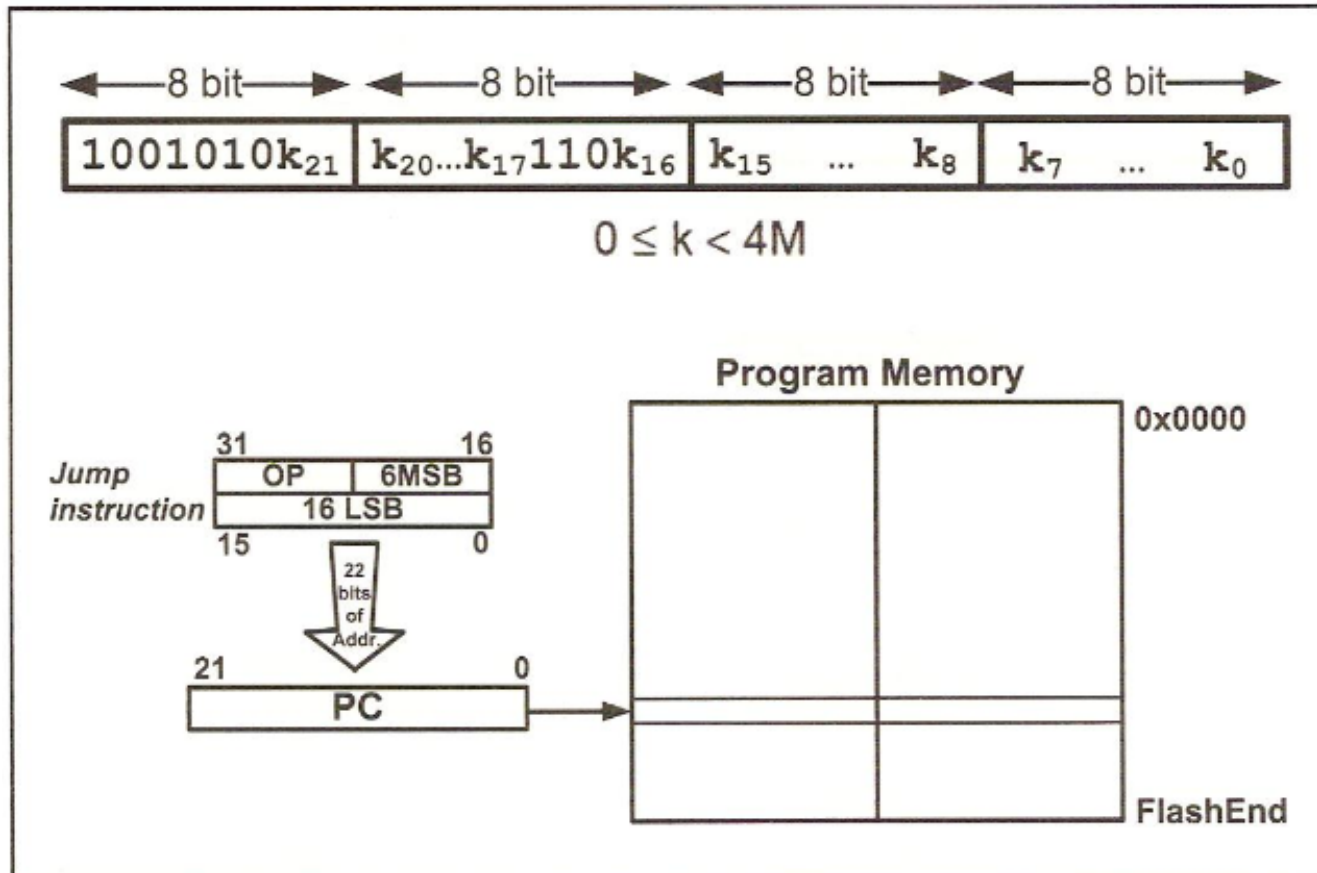


Figure 3-4. JMP Instruction

JMP AGAIN



Unconditional "short" Jump (RJMP)

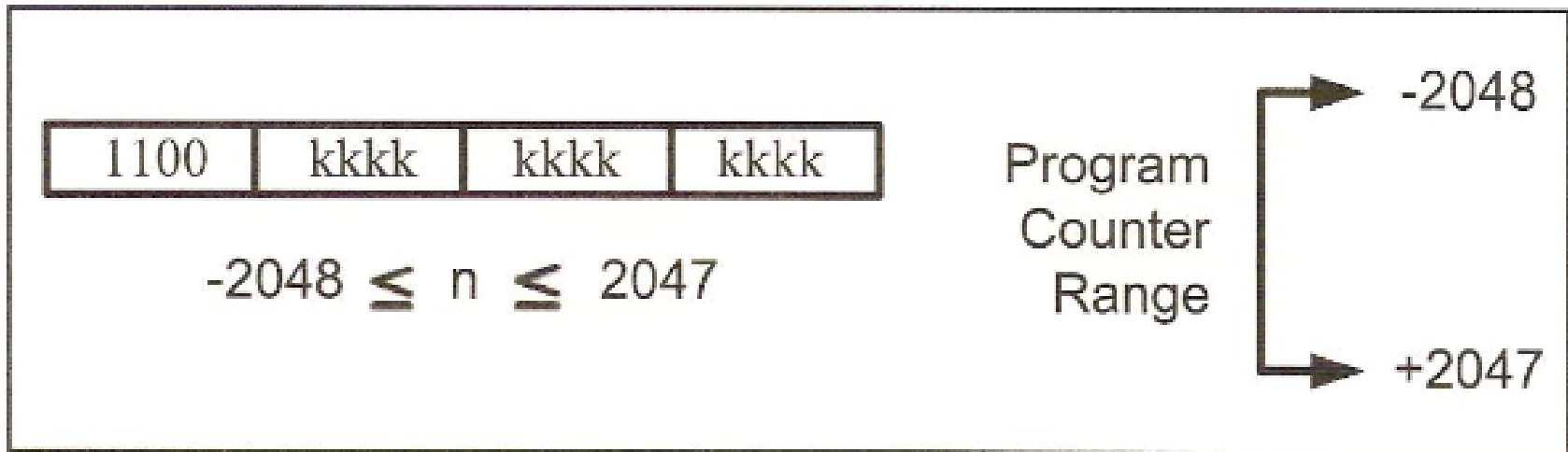


Figure 3-5. RJMP (Relative Jump) Instruction Address Range

RJMP AGAIN

Indirect Jump (IJMP)

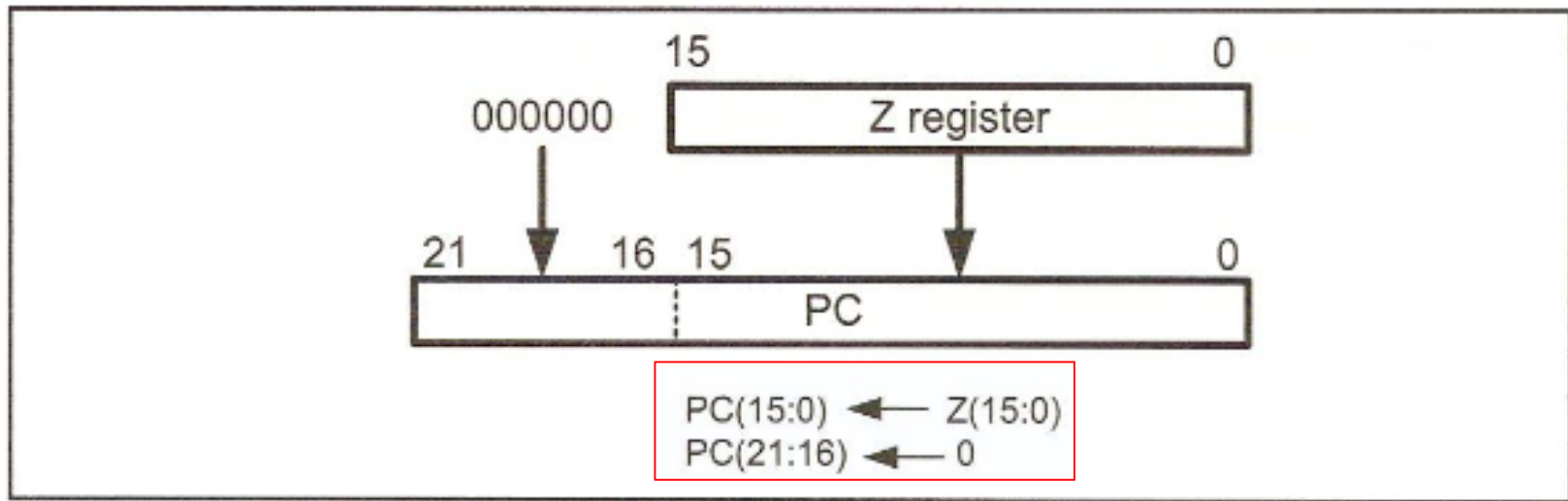


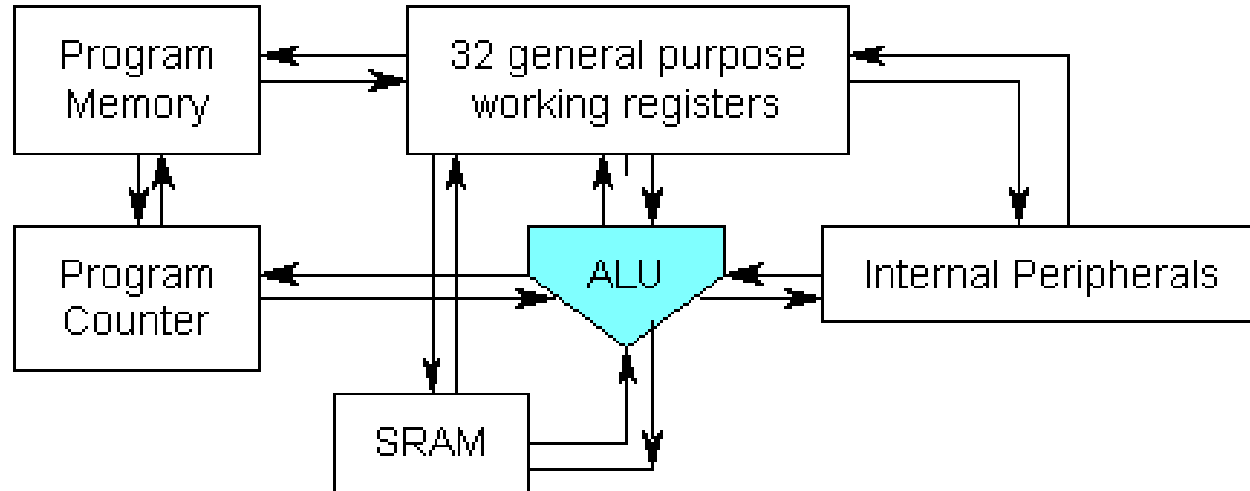
Figure 3-6. IJMP (Indirect Jump) Instruction Target Address

Rarely used !

```
LDI    R30,117 ;Z register is physical the same  
LDI    R31,47  ;as R31 and R30 combined  
IJMP
```



ALU

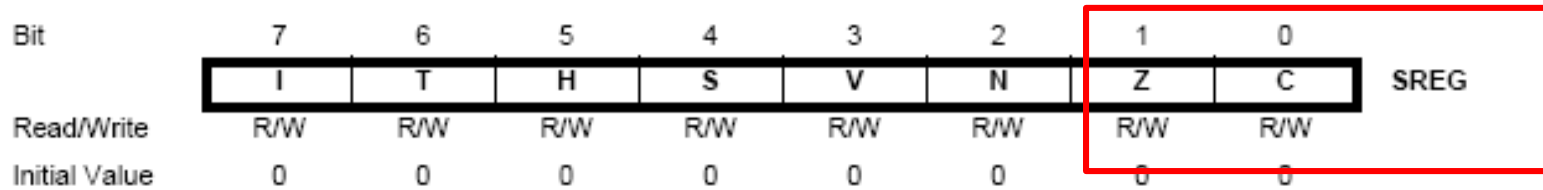


The ALU is "the calculator" of the microcontroller.

Every time we perform **an instruction involving the ALU**, some very important bits in the status register (SREG) are affected!

Each bit in SREG is called "a **flag**".

Status register SREG



- I-flag : Global interrupt enable (CLI og SEI).
- T-flag : Bit copy store (BLD og BST).
- H-flag : "Half carry" et set by "carry" from bit 3 to bit 4. Rarely used.
- S-flag : "Sign" can be used when calculating negative numbers. Depends on the N- and the V-flaget.
- V-flag : "2 complement overflow" can be used when calculating negative numbers.
- N-flag : "Negative" indicates a negative result.
- **Z-flag : "Zero"** is set (to 1), when the result is 0.
- **C-flag : "Carry"** is set, when there is a "carry" from MSB.

What instructions uses the ALU?

- Not all instructions will use the ALU ("the calculator").
- For example **MOV R12,R16** or **LDI R20,7** will not affect the status register (SREG).
- Examples of instructions affecting the SREG flags are:

INC R17

ADD R16,R17

DEC R4

Here the status flags are affected

Table 2-4: Instructions That Affect Flag Bits

| Instruction | C | Z | N | V | S | H |
|-------------|---|---|---|---|---|---|
| → ADD | X | X | X | X | X | X |
| ADC | X | X | X | X | X | X |
| ADIW | X | X | X | X | X | |
| AND | | X | X | X | X | |
| ANDI | | X | X | X | X | |
| CBR | | X | X | X | X | |
| → CLR | | X | X | X | X | |
| → COM | X | X | X | X | X | |
| → DEC | | X | X | X | X | |
| EOR | | X | X | X | X | |
| FMUL | X | X | | | | |
| → INC | | X | X | X | X | |
| LSL | X | X | X | X | | X |
| LSR | X | X | X | X | | |
| OR | | X | X | X | X | |
| ORI | | X | X | X | X | |
| ROL | X | X | X | X | | X |
| ROR | X | X | X | X | | |
| SEN | | | 1 | | | |
| SEZ | | 1 | | | | |
| → SUB | X | X | X | X | X | X |
| SUBI | X | X | X | X | X | X |
| TST | | X | X | X | X | |

Note: X can be 0 or 1. (See Chapter 5 for how to use these instructions.)

Status Register (SREG)

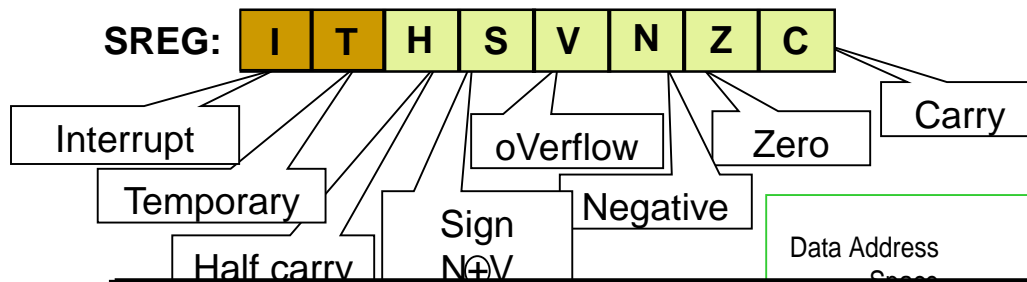


Table 2-5: AVR Branch (Jump) Instructions Using Flag Bits

| Instruction | Action |
|-------------|-----------------|
| BRLO | Branch if C = 1 |
| BRSH | Branch if C = 0 |
| BREQ | Branch if Z = 1 |
| BRNE | Branch if Z = 0 |
| BRMI | Branch if N = 1 |
| BRPL | Branch if N = 0 |
| BRVS | Branch if V = 1 |
| BRVC | Branch if V = 0 |

Example: Show the status of the C, H, and Z flags after subtraction of 0x9C from 0x9C in the following

```
LDI    R20, 0x9C
LDI    R21, 0x9C
SUB    R20, R21           ;subtract R21 from R20
```

Solution:

```

$9C   1001 1100
- $9C 1001 1100
-----
$00   0000 0000   R20 = $00
```

C = 0 because R21 is not bigger than R20 and there is no borrow from D8 bit.

Z = 1 because the R20 is zero after the subtraction.

H = 0 because there is no borrow from D4 to D3.

Conditional Jumps (uses the flags)

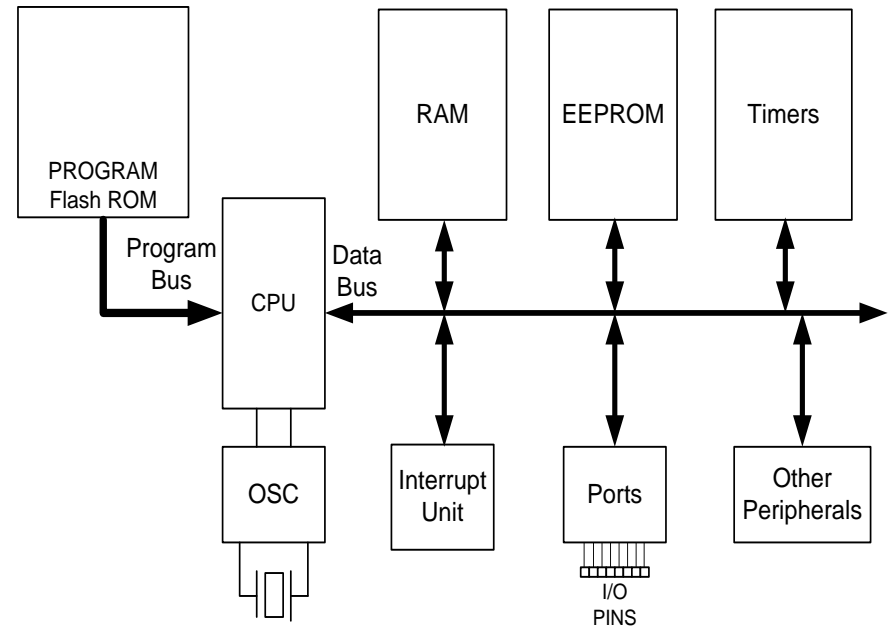
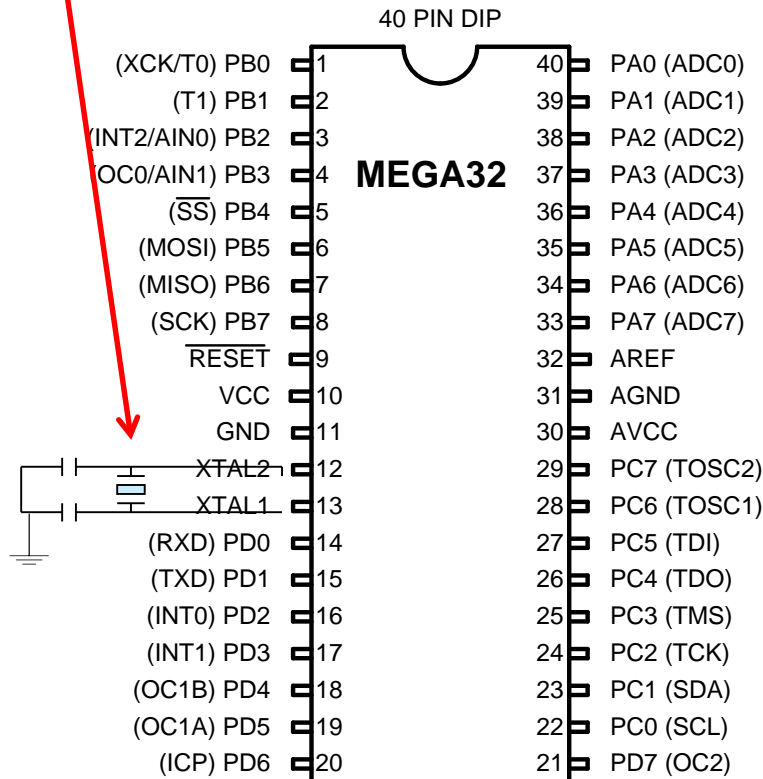
Table 2-5: AVR Branch (Jump) Instructions Using Flag Bits

| Instruction | Action |
|-------------|-------------------|
| BRLO | Branch if $C = 1$ |
| BRSH | Branch if $C = 0$ |
| BREQ | Branch if $Z = 1$ |
| BRNE | Branch if $Z = 0$ |
| BRMI | Branch if $N = 1$ |
| BRPL | Branch if $N = 0$ |
| BRVS | Branch if $V = 1$ |
| BRVC | Branch if $V = 0$ |

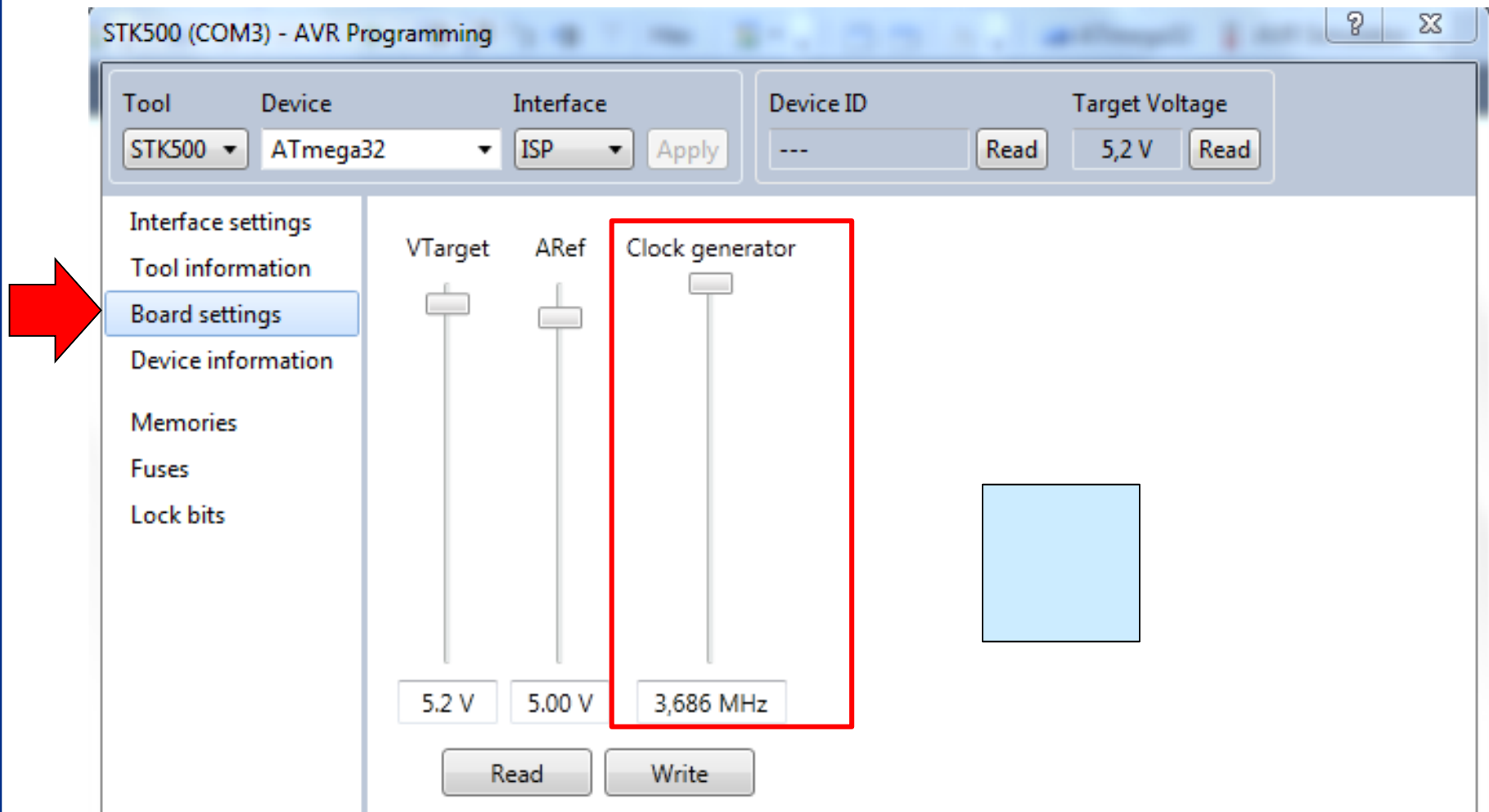


CPU clock frequency

The STK500 crystal frequency = **3,6864 MHz** !
(if correct set using Atmel Studio)

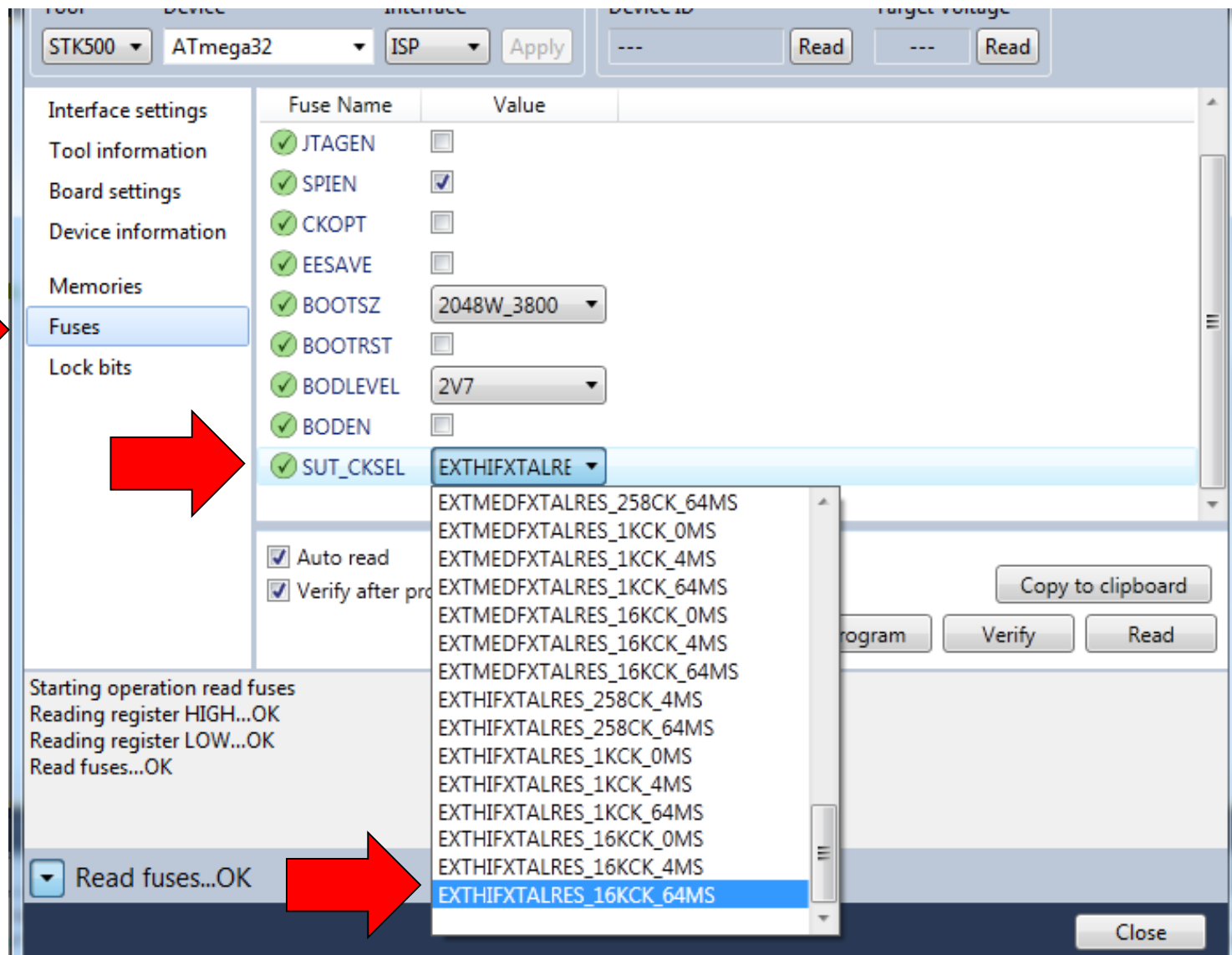


STK500: Selecting 3,6864 MHz clock



The screenshot shows the 'STK500 (COM3) - AVR Programming' window. The left sidebar contains a list of settings: 'Interface settings', 'Tool information', 'Board settings' (highlighted with a red arrow), 'Device information', 'Memories', 'Fuses', and 'Lock bits'. The main area displays three vertical sliders: 'VTarget' set to 5.2 V, 'ARef' set to 5.00 V, and 'Clock generator' set to 3,686 MHz. The 'Clock generator' section is enclosed in a red rectangle. At the top, there are dropdown menus for 'Tool' (STK500), 'Device' (ATmega32), and 'Interface' (ISP), along with an 'Apply' button. To the right, there are fields for 'Device ID' (---) and 'Target Voltage' (5,2 V), each with a 'Read' button. At the bottom, there are 'Read' and 'Write' buttons.

STK500: Selecting 3,6864 MHz clock



Arithmetic and logical instructions(1)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|----------|----------|-------------------------------|-----------------------------|-------------|--------|
| ADD | Rd,Rr | Add without Carry | $Rd = Rd + Rr$ | Z,C,N,V,H,S | 1 |
| ADC | Rd,Rr | Add with Carry | $Rd = Rd + Rr + C$ | Z,C,N,V,H,S | 1 |
| ADIW | Rd, K | Add Immediate To Word | $Rd+1 : Rd, K$ | Z,C,N,V,S | 2 |
| SUB | Rd,Rr | Subtract without Carry | $Rd = Rd - Rr$ | Z,C,N,V,H,S | 1 |
| SUBI | Rd,K8 | Subtract Immediate | $Rd = Rd - K8$ | Z,C,N,V,H,S | 1 |
| SBC | Rd,Rr | Subtract with Carry | $Rd = Rd - Rr - C$ | Z,C,N,V,H,S | 1 |
| SBCI | Rd,K8 | Subtract with Carry Immediate | $Rd = Rd - K8 - C$ | Z,C,N,V,H,S | 1 |
| AND | Rd,Rr | Logical AND | $Rd = Rd \cdot Rr$ | Z,N,V,S | 1 |
| ANDI | Rd,K8 | Logical AND with Immediate | $Rd = Rd \cdot K8$ | Z,N,V,S | 1 |
| OR | Rd,Rr | Logical OR | $Rd = Rd \vee Rr$ | Z,N,V,S | 1 |
| ORI | Rd,K8 | Logical OR with Immediate | $Rd = Rd \vee K8$ | Z,N,V,S | 1 |
| EOR | Rd,Rr | Logical Exclusive OR | $Rd = Rd \oplus Rr$ | Z,N,V,S | 1 |
| COM | Rd | One's Complement | $Rd = \$FF - Rd$ | Z,C,N,V,S | 1 |
| NEG | Rd | Two's Complement | $Rd = \$00 - Rd$ | Z,C,N,V,H,S | 1 |
| SBR | Rd,K8 | Set Bit(s) in Register | $Rd = Rd \vee K8$ | Z,C,N,V,S | 1 |
| CBR | Rd,K8 | Clear Bit(s) in Register | $Rd = Rd \cdot (\$FF - K8)$ | Z,C,N,V,S | 1 |
| INC | Rd | Increment Register | $Rd = Rd + 1$ | Z,N,V,S | 1 |
| DEC | Rd | Decrement Register | $Rd = Rd - 1$ | Z,N,V,S | 1 |
| TST | Rd | Test for Zero or Negative | $Rd = Rd \cdot Rd$ | Z,C,N,V,S | 1 |
| CLR | Rd | Clear Register | $Rd = 0$ | Z,C,N,V,S | 1 |
| SER | Rd | Set Register | $Rd = \$FF$ | None | 1 |

Arithmetic and logical instructions(2)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|------------------------|-------------------------|--|---------------------------|-----------|--------|
| SBIW | RdI, K6 | Subtract Immediate from Word | $RdH:RdL = RdH:RdL - K_6$ | Z,C,N,V,S | 2 |
| MUL | Rd, Rr | Multiply Unsigned | $R1:R0 = Rd * Rr$ | Z,C | 2 |
| MULS | Rd, Rr | Multiply Signed | $R1:R0 = Rd * Rr$ | Z,C | 2 |
| MULSU | Rd, Rr | Multiply Signed with Unsigned | $R1:R0 = Rd * Rr$ | Z,C | 2 |
| FMUL | Rd, Rr | Fractional Multiply Unsigned | $R1:R0 = (Rd * Rr) \ll 1$ | Z,C | 2 |
| FMULS | Rd, Rr | Fractional Multiply Signed | $R1:R0 = (Rd * Rr) \ll 1$ | Z,C | 2 |
| FMULSU | Rd, Rr | Fractional Multiply Signed with Unsigned | $R1:R0 = (Rd * Rr) \ll 1$ | Z,C | 2 |

Branch instructions (1)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|-----------------------|---------------|-------------------------------------|--|------------------|--------|
| RJMP | <u>k</u> | Relative Jump | $PC = PC + k + 1$ | None | 2 |
| IJMP | None | Indirect Jump to (<u>Z</u>) | $PC = Z$ | None | 2 |
| JMP | <u>k</u> | Jump | $PC = k$ | None | 3 |
| RCALL | <u>k</u> | Relative Call Subroutine | $STACK = PC + 1, PC = PC + k + 1$ | None | 3/4* |
| ICALL | None | Indirect Call to (<u>Z</u>) | $STACK = PC + 1, PC = Z$ | None | 3/4* |
| CALL | <u>k</u> | Call Subroutine | $STACK = PC + 2, PC = k$ | None | 4/5* |
| RET | None | Subroutine Return | $PC = STACK$ | None | 4/5* |
| RETI | None | Interrupt Return | $PC = STACK$ | I | 4/5* |
| CPSE | <u>Rd, Rr</u> | Compare, Skip if equal | if ($Rd == Rr$) $PC = PC + 2$ or 3 | None | 1/2/3 |
| CP | <u>Rd, Rr</u> | Compare | $Rd - Rr$ | Z, C, N, V, H, S | 1 |
| CPC | <u>Rd, Rr</u> | Compare with Carry | $Rd - Rr - C$ | Z, C, N, V, H, S | 1 |
| CPI | <u>Rd, K8</u> | Compare with Immediate | $Rd - K$ | Z, C, N, V, H, S | 1 |
| SBRC | <u>Rr, b</u> | Skip if bit in register cleared | if ($Rr(b) == 0$) $PC = PC + 2$ or 3 | None | 1/2/3 |
| SBRS | <u>Rr, b</u> | Skip if bit in register set | if ($Rr(b) == 1$) $PC = PC + 2$ or 3 | None | 1/2/3 |
| SBIC | <u>P, b</u> | Skip if bit in I/O register cleared | if ($I/O(P, b) == 0$) $PC = PC + 2$ or 3 | None | 1/2/3 |
| SBIS | <u>P, b</u> | Skip if bit in I/O register set | if ($I/O(P, b) == 1$) $PC = PC + 2$ or 3 | None | 1/2/3 |
| BRBC | <u>s, k</u> | Branch if Status flag cleared | if ($SREG(s) == 0$) $PC = PC + k + 1$ | None | 1/2 |
| BRBS | <u>s, k</u> | Branch if Status flag set | if ($SREG(s) == 1$) $PC = PC + k + 1$ | None | 1/2 |

Branch instructions (2)



| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|----------------------|----------|--|--------------------------|-------|--------|
| BREQ | <u>k</u> | Branch if equal | if(Z==1) PC = PC + k + 1 | None | 1/2 |
| BRNE | <u>k</u> | Branch if not equal | if(Z==0) PC = PC + k + 1 | None | 1/2 |
| BRCS | <u>k</u> | Branch if carry set | if(C==1) PC = PC + k + 1 | None | 1/2 |
| BRCC | <u>k</u> | Branch if carry cleared | if(C==0) PC = PC + k + 1 | None | 1/2 |
| BRSH | <u>k</u> | Branch if same or higher | if(C==0) PC = PC + k + 1 | None | 1/2 |
| BRLO | <u>k</u> | Branch if lower | if(C==1) PC = PC + k + 1 | None | 1/2 |
| BRMI | <u>k</u> | Branch if minus | if(N==1) PC = PC + k + 1 | None | 1/2 |
| BRPL | <u>k</u> | Branch if plus | if(N==0) PC = PC + k + 1 | None | 1/2 |
| BRGE | <u>k</u> | Branch if greater than or equal (signed) | if(S==0) PC = PC + k + 1 | None | 1/2 |
| BRLT | <u>k</u> | Branch if less than (signed) | if(S==1) PC = PC + k + 1 | None | 1/2 |
| BRHS | <u>k</u> | Branch if half carry flag set | if(H==1) PC = PC + k + 1 | None | 1/2 |
| BRHC | <u>k</u> | Branch if half carry flag cleared | if(H==0) PC = PC + k + 1 | None | 1/2 |
| BRTS | <u>k</u> | Branch if T flag set | if(T==1) PC = PC + k + 1 | None | 1/2 |
| BRTC | <u>k</u> | Branch if T flag cleared | if(T==0) PC = PC + k + 1 | None | 1/2 |
| BRVS | <u>k</u> | Branch if overflow flag set | if(V==1) PC = PC + k + 1 | None | 1/2 |
| BRVC | <u>k</u> | Branch if overflow flag cleared | if(V==0) PC = PC + k + 1 | None | 1/2 |
| BRIE | <u>k</u> | Branch if interrupt enabled | if(I==1) PC = PC + k + 1 | None | 1/2 |
| BRID | <u>k</u> | Branch if interrupt disabled | if(I==0) PC = PC + k + 1 | None | 1/2 |



Data transfer instructions (1)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|----------------------|------------------------|----------------------------------|-------------------------------|-------|--------|
| MOV | Rd,Rr | Copy register | $Rd = Rr$ | None | 1 |
| MOVW | Rd,Rr | Copy register pair | $Rd+1:Rd = Rr+1:Rr, r,d$ even | None | 1 |
| LDI | Rd,K8 | Load Immediate | $Rd = K$ | None | 1 |
| LDS | Rd,k | Load Direct | $Rd = (k)$ | None | 2* |
| LD | Rd,X | Load Indirect | $Rd = (X)$ | None | 2* |
| LD | Rd,X+ | Load Indirect and Post-Increment | $Rd = (X), X=X+1$ | None | 2* |
| LD | Rd,-X | Load Indirect and Pre-Decrement | $X=X-1, Rd = (X)$ | None | 2* |
| LD | Rd,Y | Load Indirect | $Rd = (Y)$ | None | 2* |
| LD | Rd,Y+ | Load Indirect and Post-Increment | $Rd = (Y), Y=Y+1$ | None | 2* |
| LD | Rd,-Y | Load Indirect and Pre-Decrement | $Y=Y-1, Rd = (Y)$ | None | 2* |
| LDD | Rd,Y+q | Load Indirect with displacement | $Rd = (Y+q)$ | None | 2* |
| LD | Rd,Z | Load Indirect | $Rd = (Z)$ | None | 2* |
| LD | Rd,Z+ | Load Indirect and Post-Increment | $Rd = (Z), Z=Z+1$ | None | 2* |
| LD | Rd,-Z | Load Indirect and Pre-Decrement | $Z=Z-1, Rd = (Z)$ | None | 2* |
| LDD | Rd,Z+q | Load Indirect with displacement | $Rd = (Z+q)$ | None | 2* |

Data transfer instructions (2)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|--|------------------------|--|-------------------|-------|--------|
|  STS | k,Rr | Store Direct | $(k) = Rr$ | None | 2* |
| ST | X,Rr | Store Indirect | $(X) = Rr$ | None | 2* |
| ST | X+,Rr | Store Indirect and Post-Increment | $(X) = Rr, X=X+1$ | None | 2* |
| ST | -X,Rr | Store Indirect and Pre-Decrement | $X=X-1, (X)=Rr$ | None | 2* |
| ST | Y,Rr | Store Indirect | $(Y) = Rr$ | None | 2* |
| ST | Y+,Rr | Store Indirect and Post-Increment | $(Y) = Rr, Y=Y+1$ | None | 2 |
| ST | -Y,Rr | Store Indirect and Pre-Decrement | $Y=Y-1, (Y) = Rr$ | None | 2 |
| ST | Y+q,Rr | Store Indirect with displacement | $(Y+q) = Rr$ | None | 2 |
| ST | Z,Rr | Store Indirect | $(Z) = Rr$ | None | 2 |
| ST | Z+,Rr | Store Indirect and Post-Increment | $(Z) = Rr, Z=Z+1$ | None | 2 |
| ST | -Z,Rr | Store Indirect and Pre-Decrement | $Z=Z-1, (Z) = Rr$ | None | 2 |
| ST | Z+q,Rr | Store Indirect with displacement | $(Z+q) = Rr$ | None | 2 |
| LPM | None | Load Program Memory | $R0 = (Z)$ | None | 3 |
| LPM | Rd,Z | Load Program Memory | $Rd = (Z)$ | None | 3 |
| LPM | Rd,Z+ | Load Program Memory and Post-Increment | $Rd = (Z), Z=Z+1$ | None | 3 |
| SPM | None | Store Program Memory | $(Z) = R1:R0$ | None | - |
|  IN | Rd,P | In Port | $Rd = P$ | None | 1 |
| OUT | P,Rr | Out Port | $P = Rr$ | None | 1 |
| PUSH | Rr | Push register on Stack | $STACK = Rr$ | None | 2 |
| POP | Rd | Pop register from Stack | $Rd = STACK$ | None | 2 |

Bit- and Bit Test instructions (1)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|----------------------|----------------------|------------------------------|---|-------------|--------|
| LSL | Rd | Logical shift left | $Rd(n+1)=Rd(n)$, $Rd(0)=0$, $C=Rd(7)$ | Z,C,N,V,H,S | 1 |
| LSR | Rd | Logical shift right | $Rd(n)=Rd(n+1)$, $Rd(7)=0$, $C=Rd(0)$ | Z,C,N,V,S | 1 |
| ROL | Rd | Rotate left through carry | $Rd(0)=C$, $Rd(n+1)=Rd(n)$, $C=Rd(7)$ | Z,C,N,V,H,S | 1 |
| ROR | Rd | Rotate right through carry | $Rd(7)=C$, $Rd(n)=Rd(n+1)$, $C=Rd(0)$ | Z,C,N,V,S | 1 |
| ASR | Rd | Arithmetic shift right | $Rd(n)=Rd(n+1)$, $n=0,\dots,6$ | Z,C,N,V,S | 1 |
| SWAP | Rd | Swap nibbles | $Rd(3..0) = Rd(7..4)$, $Rd(7..4) = Rd(3..0)$ | None | 1 |
| BSET | s | Set flag | $SREG(s) = 1$ | SREG(s) | 1 |
| BCLR | s | Clear flag | $SREG(s) = 0$ | SREG(s) | 1 |
| SBI | P,b | Set bit in I/O register | $I/O(P,b) = 1$ | None | 2 |
| CBI | P,b | Clear bit in I/O register | $I/O(P,b) = 0$ | None | 2 |
| BST | Rr,b | Bit store from register to T | $T = Rr(b)$ | T | 1 |
| BLD | Rd,b | Bit load from register to T | $Rd(b) = T$ | None | 1 |



Bit- and Bit Test instructions (2)

| Mnemonic | Operands | Description | Operation | Flags | Cycles |
|-----------------------|----------|-----------------------|------------------------|-------|--------|
| SEC | None | Set carry flag | C = 1 | C | 1 |
| CLC | None | Clear carry flag | C = 0 | C | 1 |
| SEN | None | Set negative flag | N = 1 | N | 1 |
| CLN | None | Clear negative flag | N = 0 | N | 1 |
| SEZ | None | Set zero flag | Z = 1 | Z | 1 |
| CLZ | None | Clear zero flag | Z = 0 | Z | 1 |
| SEI | None | Set interrupt flag | I = 1 | I | 1 |
| CLI | None | Clear interrupt flag | I = 0 | I | 1 |
| SES | None | Set signed flag | S = 1 | S | 1 |
| CLN | None | Clear signed flag | S = 0 | S | 1 |
| SEV | None | Set overflow flag | V = 1 | V | 1 |
| CLV | None | Clear overflow flag | V = 0 | V | 1 |
| SET | None | Set T-flag | T = 1 | T | 1 |
| CLT | None | Clear T-flag | T = 0 | T | 1 |
| SEH | None | Set half carry flag | H = 1 | H | 1 |
| CLH | None | Clear half carry flag | H = 0 | H | 1 |
| NOP | None | No operation | None | None | 1 |
| SLEEP | None | Sleep | See instruction manual | None | 1 |
| WDR | None | Watchdog Reset | See instruction manual | None | 1 |



NOP (No Operation)

Description:

This instruction performs a single cycle No Operation.

Operation:

(i) No

Syntax:

(i) NOP

Operands:

None

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

| | | | |
|------|------|------|------|
| 0000 | 0000 | 0000 | 0000 |
|------|------|------|------|

Status Register (SREG) and Boolean Formula:

| I | T | H | S | V | N | Z | C |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |

Example:

```
clr    r16    ; Clear r16
ser    r17    ; Set r17
out    $18,r16 ; Write zeros to Port B
nop                    ; Wait (do nothing)
out    $18,r17 ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

Time delay: Example

| | | machine cycle |
|-----|----------|--------------------------|
| LDI | R16, 19 | 1 |
| LDI | R20, 95 | 1 |
| LDI | R21, 5 | 1 |
| ADD | R16, R20 | 1 |
| ADD | R16, R21 | <u>1</u> |
| | | 5 |

Time delay: Example

| | | | machine cycle | |
|--------|------|----------|---------------|----------------|
| | LDI | R16, 100 | 1 | |
| AGAIN: | ADD | R17,R16 | 1 | *100 |
| | DEC | R16 | 1 | *100 |
| | BRNE | AGAIN | 1 | *100 |
| | | | 1/2 | |
| | | | | Branch penalty |

Time delay: Example

| | | <u>machine cycle</u> | |
|--------|------|----------------------|------------|
| | LDI | R16, 50 | 1 |
| AGAIN: | NOP | | 1 *50 |
| | NOP | | 1 *50 |
| | DEC | R16 | 1 *50 |
| | BRNE | AGAIN | 1 *50 |
| | | | <u>1/2</u> |

Time delay: Example

| | | | machine cycle |
|-----|------|---------|--------------------------|
| | LDI | R17, 20 | |
| L1: | LDI | R16, 50 | 1 |
| L2: | NOP | | 1 *20 |
| | NOP | | 1 *20 * 50 |
| | DEC | R16 | 1 *20 * 50 |
| | BRNE | L2 | 1 *20 * 50 |
| | DEC | R17 | 1/2 *20 * 50 |
| | BRNE | L1 | 1 *20 |
| | | | <u>1/2</u> *20 |

Code from LAB2

```
;***** DELAY *****  
DELAY:  
    CLR    R17  
    LDI    R18,200    ;Large number = long delay  
AGAIN:  
    DEC    R17  
    BRNE   AGAIN  
    DEC    R18  
    BRNE   AGAIN  
    RET  
;*****
```

EXERCISE: What is the execution time for this function?

Answer at "socrative.com": Room = MSYS

Code from LAB1

```
;***** DISPLAY R16 *****  
;***** AND DELAY *****  
DISP_AND_DELAY:  
    MOV    R17,R16  
    COM    R17  
    OUT    PORTB,R17  
    CLR    R17  
    CLR    R18  
    LDI    R19,10  
AGAIN:  
    DEC    R17  
    BRNE   AGAIN  
    DEC    R18  
    BRNE   AGAIN  
    DEC    R19  
    BRNE   AGAIN  
    RET  
;*****
```

EXERCISE: What is the execution time for this function?

Answer at "socrative.com": Room = MSYS

Simulator Stop Watch (and Cycle Counter)

| Processor | | |
|-----------------|-----------------|--|
| Name | Value | |
| Program Counter | 0x00000022 | |
| Stack Pointer | 0x0000085B | |
| X Register | 0x0000 | |
| Y Register | 0x0000 | |
| Z Register | 0x0000 | |
| Status Register | I T H S V N Z C | |
| Cycle Counter | 21 | |
| Frequency | 3,686 MHz | |
| Stop Watch | 5,70 μ s | |
| + Registers | | |

ADD (Add without Carry)

Description:

Adds two registers without the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr$

Syntax:

(i) `ADD Rd,Rr`

Operands:

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

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

| | | | |
|------|------|------|------|
| 0000 | 11rd | dddd | rrrr |
|------|------|------|------|

Example:

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

`add r28,r28 ; Add r28 to itself (r28=r28+r28)`

ADC (Add with Carry)

Description:

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr + C$

Syntax:

(i) ADC Rd,Rr

Operands:

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

Program Counter:

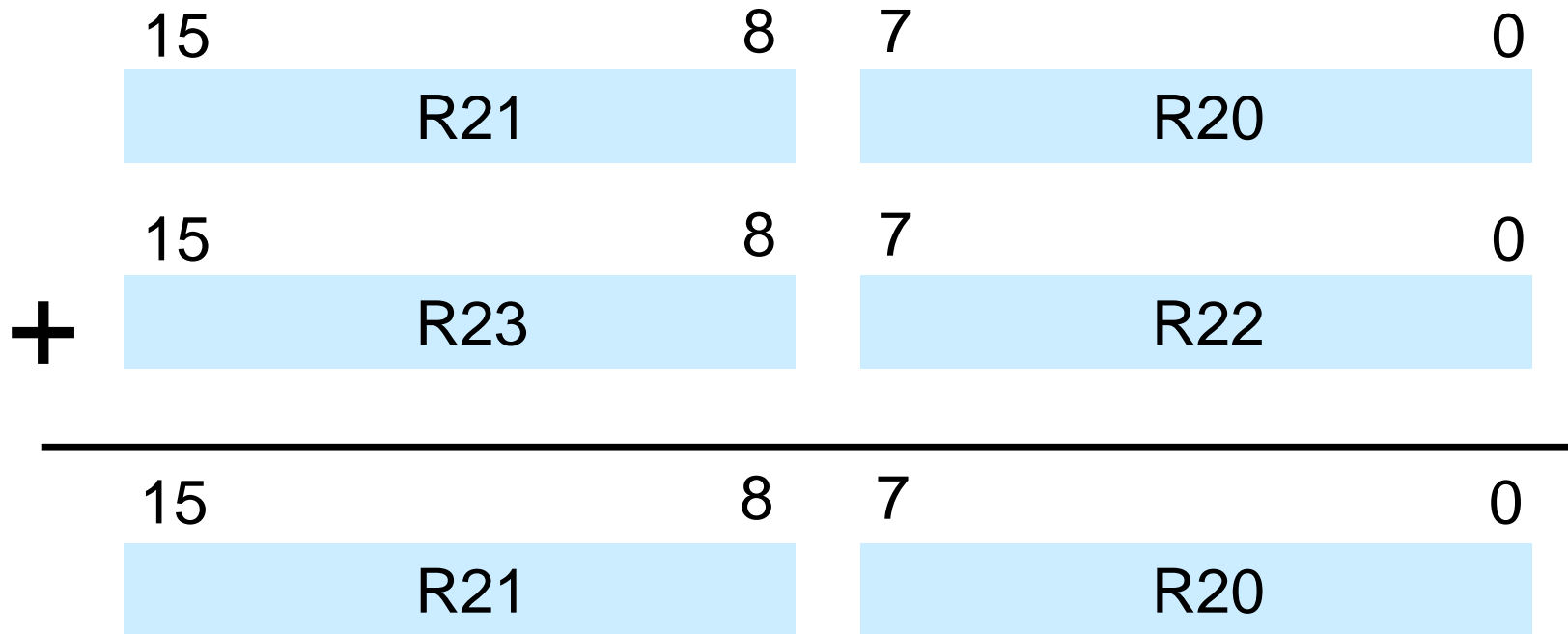
$PC \leftarrow PC + 1$

Example:

```
                ; Add R1:R0 to R3:R2
add    r2,r0    ; Add low byte
adc    r3,r1    ; Add with carry high byte
```

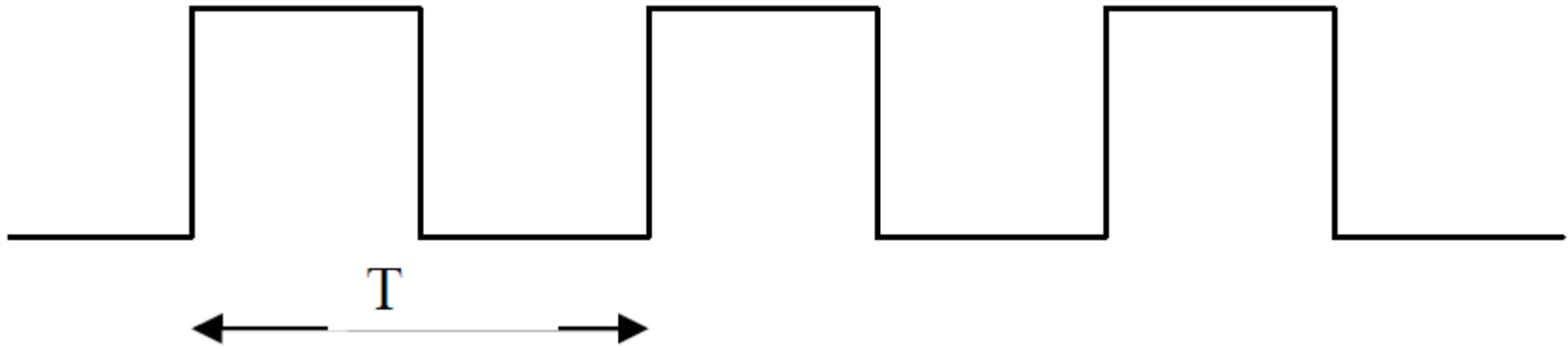
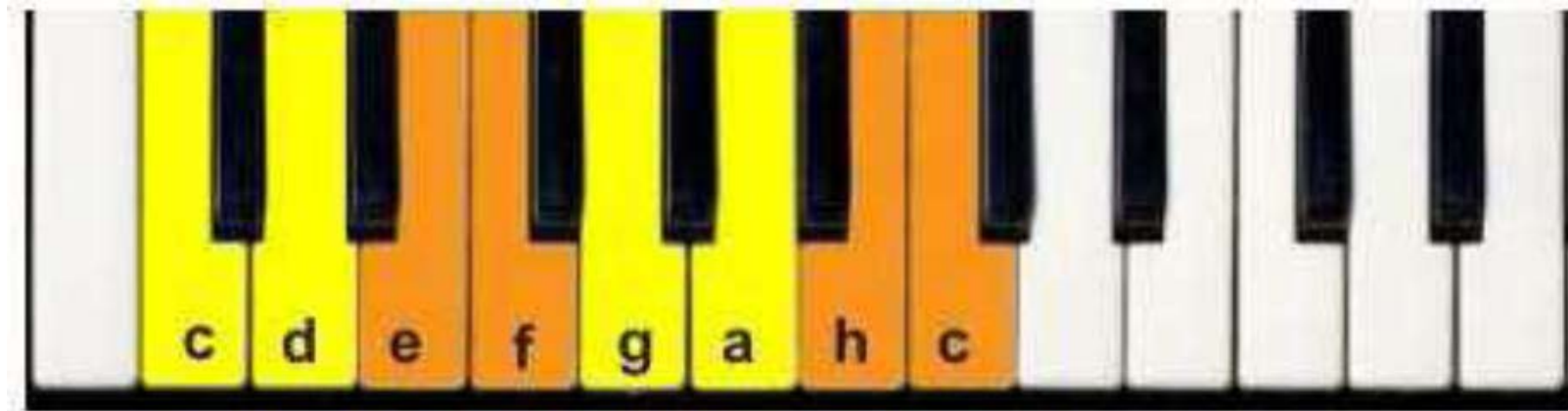
Adding 16 bit numbers

TASK: Write code to add two 16 bit numbers.



Hint: Make use of the instructions ADD and ADC.

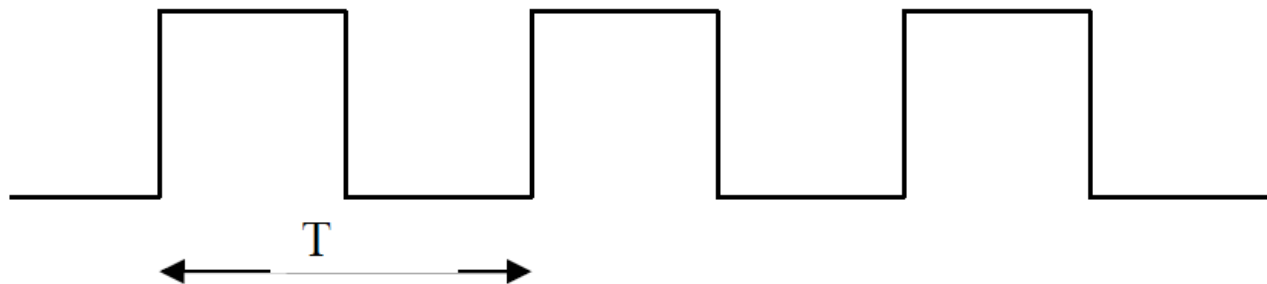
Explaining LAB3



LAB3

| Tone | Frequency | $T = 1/f$ | $T/2$ | $(T/2)/(4\mu s)$ |
|----------|------------|--------------|-------------|------------------|
| c | 523,25 Hz | 1911 μs | 956 μs | 239 |
| D | 587,33 Hz | 1792 μs | 851 μs | 213 |
| E | 659,26 Hz | 1517 μs | 758 μs | 190 |
| F | 698,46 Hz | 1432 μs | 716 μs | 179 |
| G | 783,99 Hz | 1276 μs | 638 μs | 160 |
| A | 880,00 Hz | 1136 μs | 568 μs | 142 |
| H | 987,77 Hz | 1012 μs | 506 μs | 127 |
| C | 1046,50 Hz | 956 μs | 478 μs | 120 |

Table 1: Frequencies + calculations for a C major scale.



LAB3

```
;***** PROGRAM LOOP *****  
    LDI    R20,239    ;C  
    CALL   TONE  
    LDI    R20,213    ;D  
    CALL   TONE  
    LDI    R20,190    ;E  
    CALL   TONE  
    LDI    R20,179    ;F  
    CALL   TONE  
    LDI    R20,160    ;G  
    CALL   TONE  
    LDI    R20,142    ;A  
    CALL   TONE  
    LDI    R20,127    ;H  
    CALL   TONE  
    LDI    R20,120    ;C  
    CALL   TONE  
HERE:  
    JMP    HERE
```

LAB3

```
;*****  
;*****  DELAY (R18*4us)  *****  
;*****
```

DELAY:

```
    LDI    R17,xxx    ;<----- xxx has to be calculated
```

AGAIN:

```
    DEC    R17  
    BRNE   AGAIN  
    DEC    R18  
    BRNE   DELAY  
    RET
```

```
;*****  
;****  PLAYES 1 HALF PERIOD  *****  
;****  250 HALF PERIODES      *****  
;****  T/2 = (R20)*4 us        *****  
;*****
```

TONE:

```
                ;<----- Write this code
```

```
    RET
```

```
;*****
```

End of lesson 6

