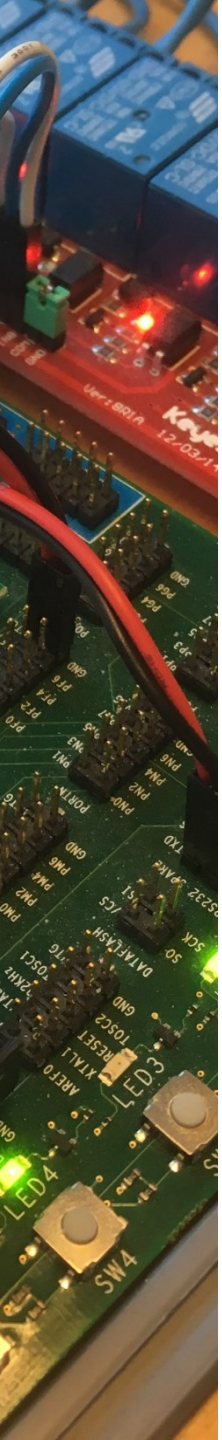
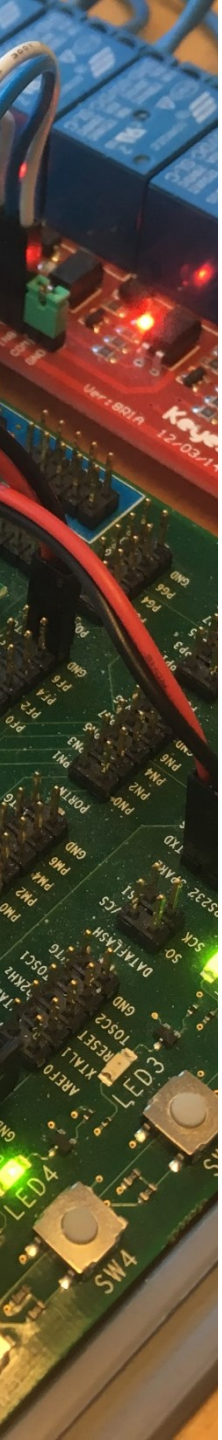
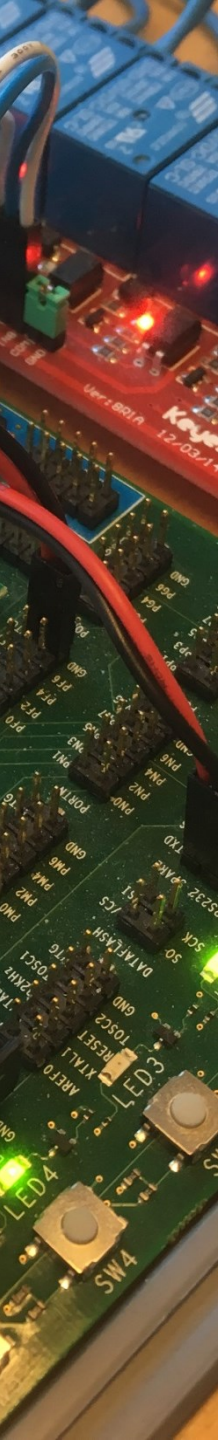


1DT301, Computer Technology
Lecture #2,
Wednesday 2019-09-04
Program example,
some instructions,
STK600.
Introduction to Lab 1.

STK 600



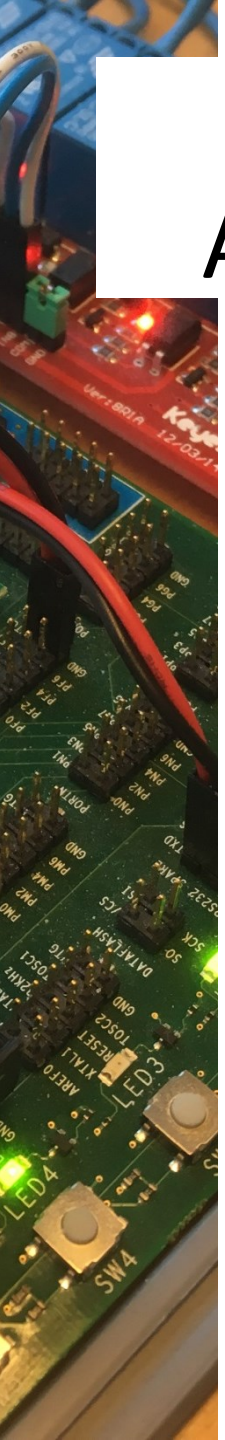


Documentation:

STK600, ATMega2560:

- doc2549_ATmega2560.pdf – ATmega2560, - 8-bit Atmel Microcontroller (STK600)
- Instruction Set.pdf

- (AVR An Introductory Course, John Morton)
- (Mikroprocessorteknik, Per Foyer)



doc2549_ATmega2560
ATMega2560, 8-bit Microcontroller

Features

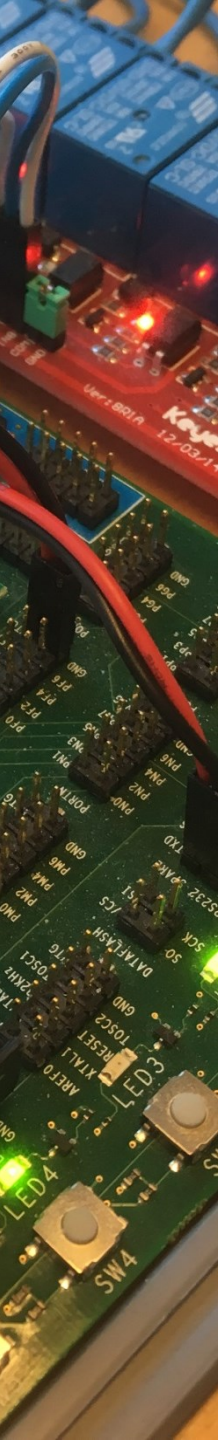
- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix® acquisition
  - Up to 64 sense channels
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green

ATMEL

8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash

ATmega640/V
ATmega1280/V
ATmega1281/V
ATmega2560/V
ATmega2561/V

Preliminary



8-bit AVR Instruction Set

Instruction Set Nomenclature

Status Register (SREG)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address

ATMEL®

8-bit AVR® Instruction Set

Architecture

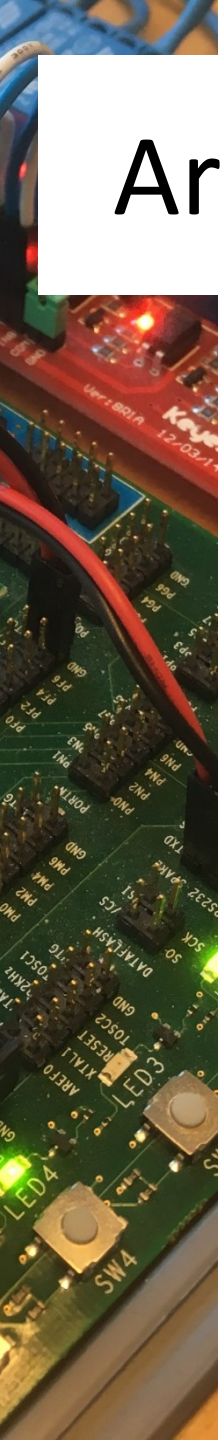
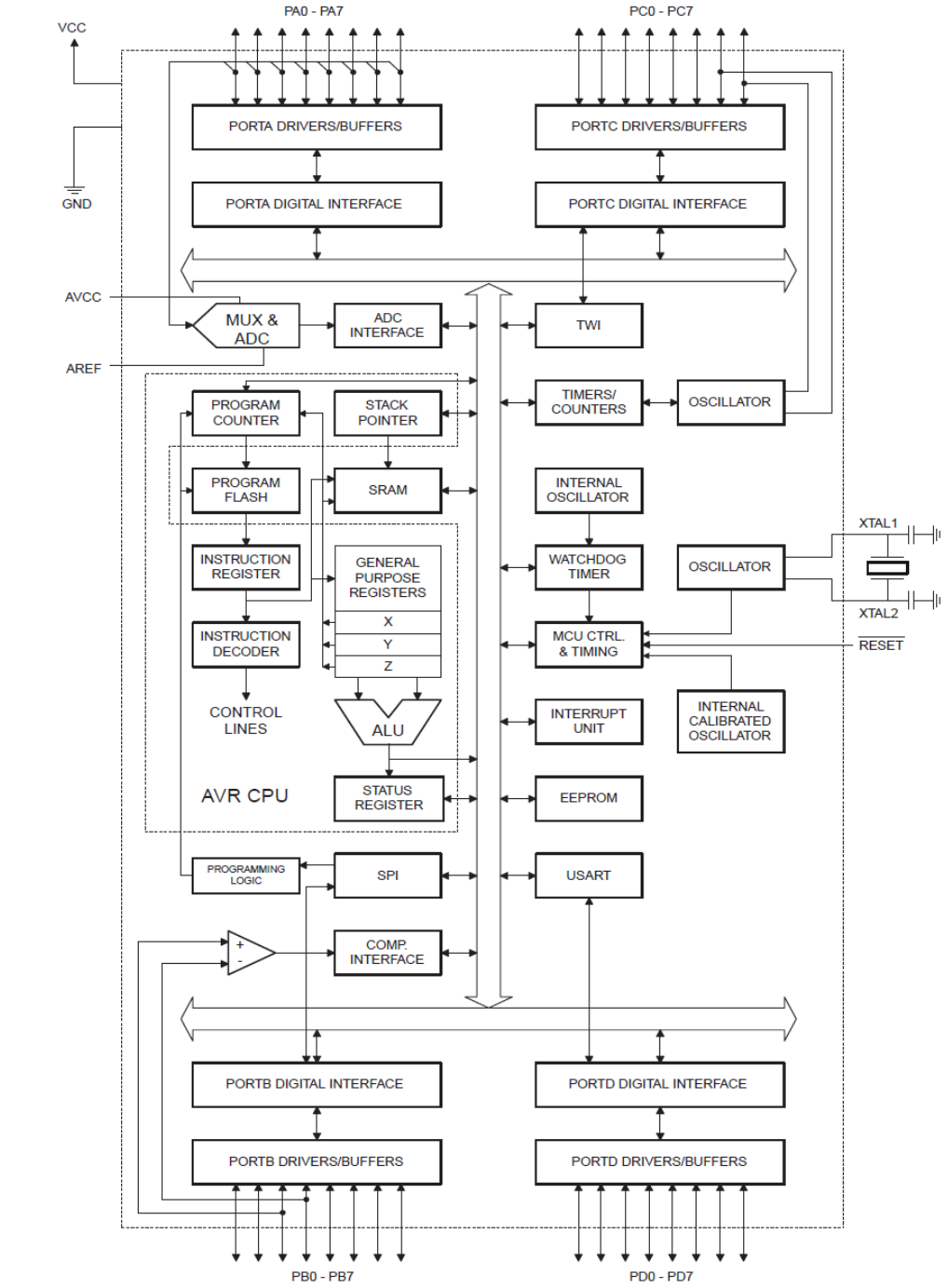
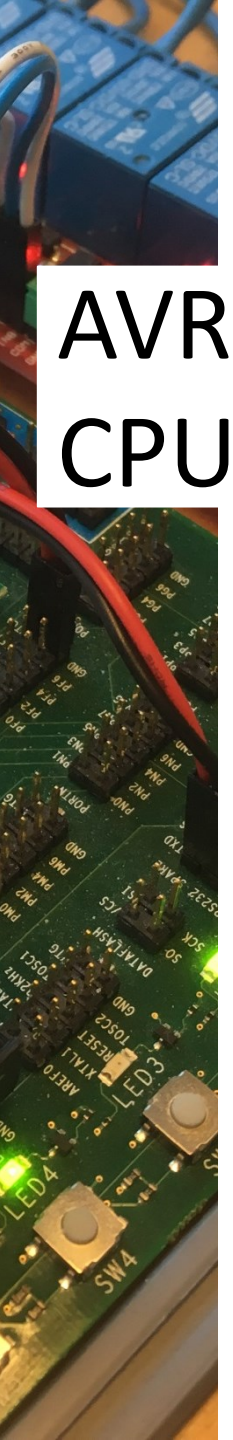


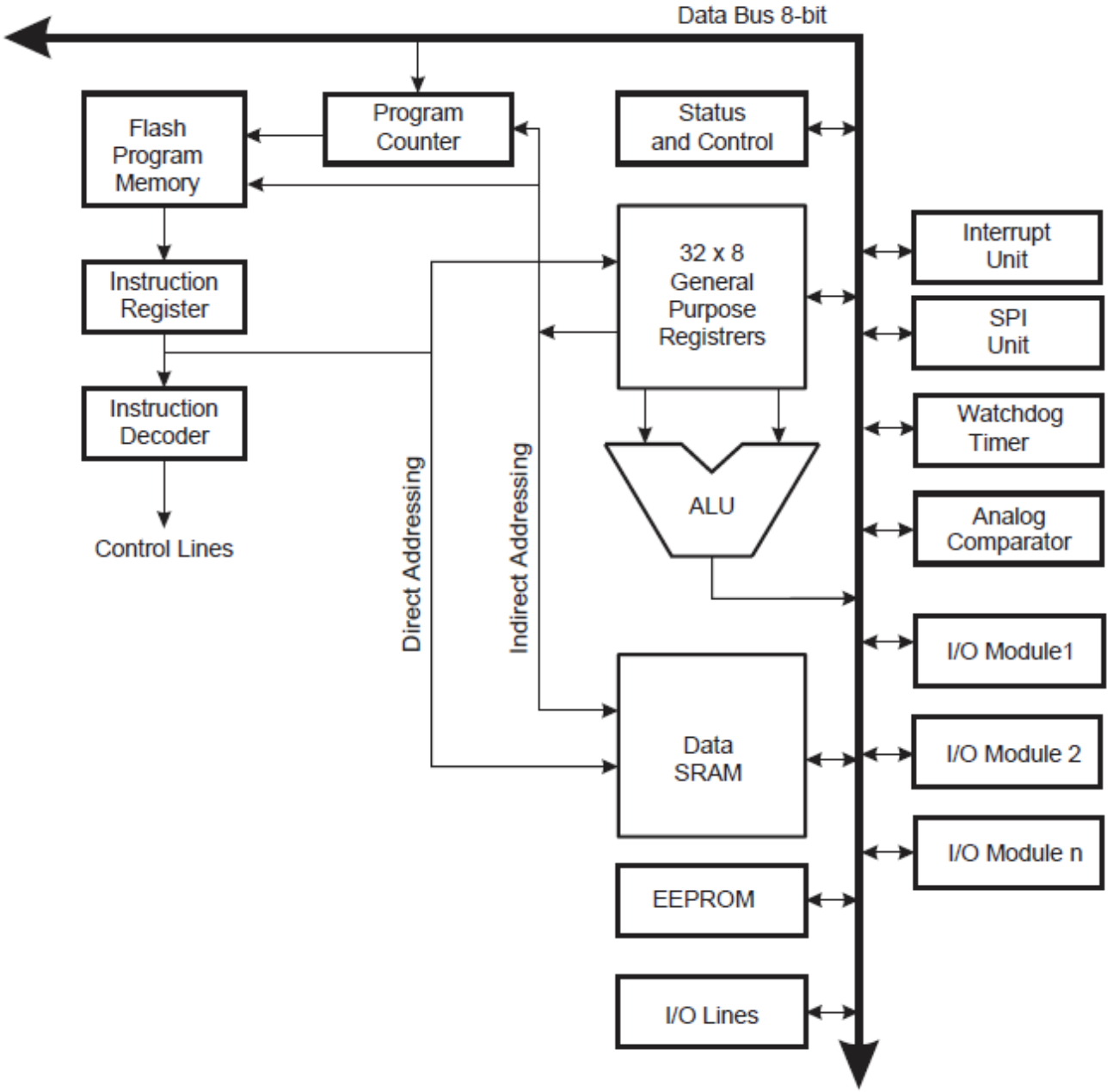
Figure 2. Block Diagram





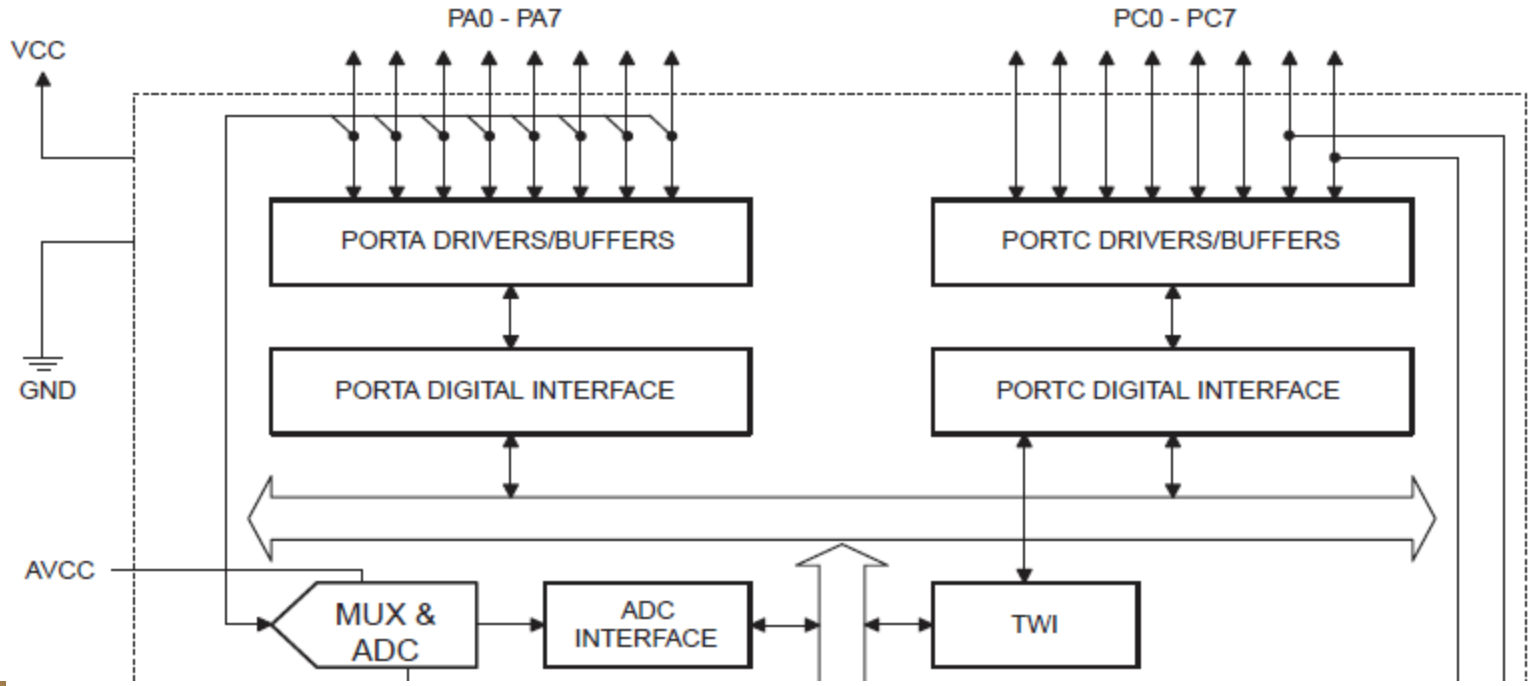
AVR CPU Core

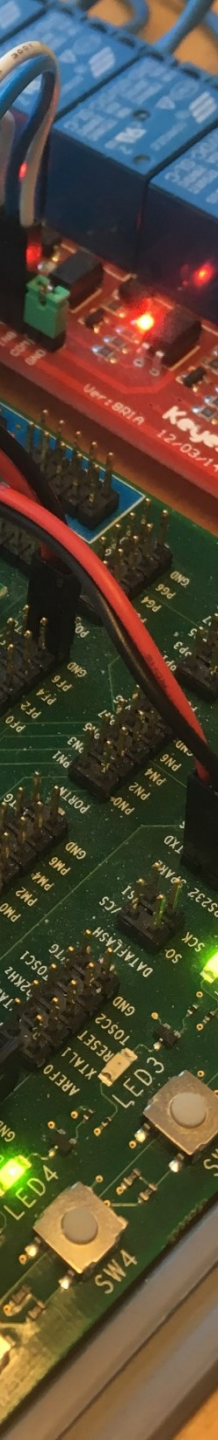
Figure 5. Block Diagram of the AVR MCU Architecture



I/O Ports

Figure 2. Block Diagram





doc2466_ATmega16, page 66

Register Description for I/O Ports

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
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	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
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	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
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	

Arduino, example:

Example

Sets pin 13 to the same value as pin 7, declared as an input.

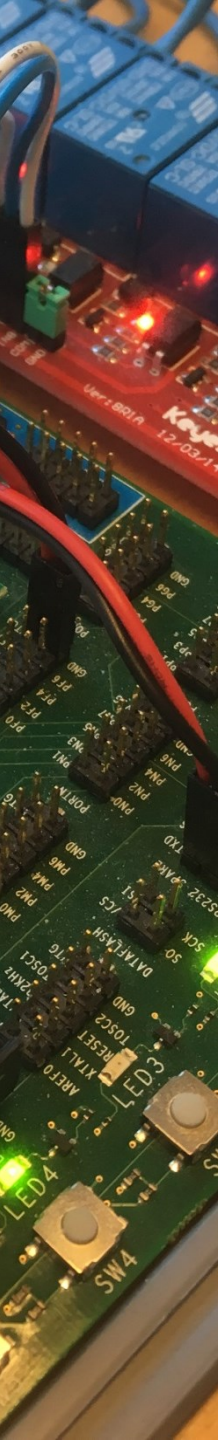
```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7;   // pushbutton connected to digital pin 7
int val = 0;     // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin 13 as output
  pinMode(inPin, INPUT);        // sets the digital pin 7 as input
}

void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);    // sets the LED to the button's value
}
```

Digital input/outputs
has to be configurded, in or out

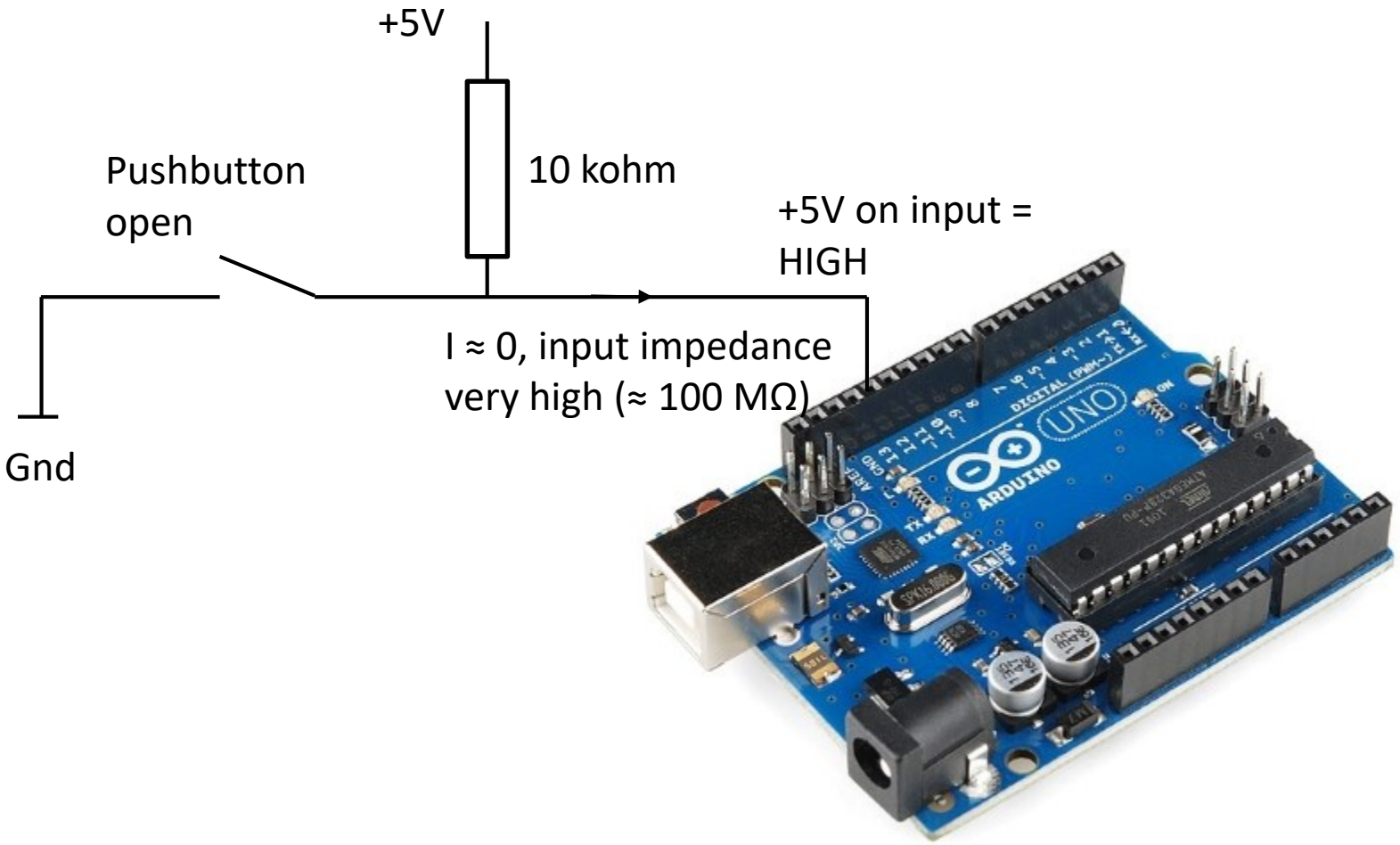
Note

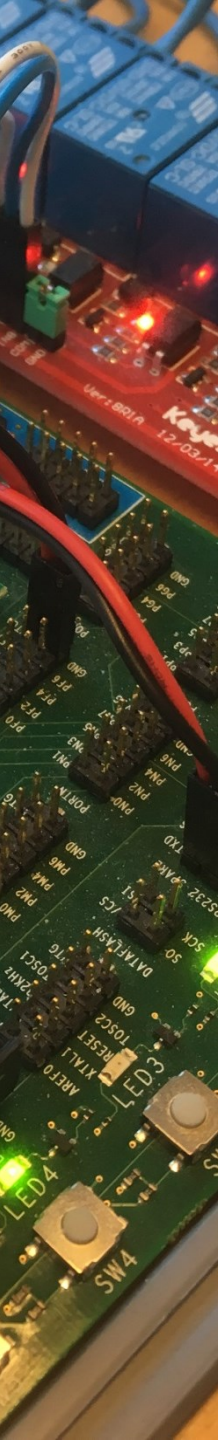
If the pin isn't connected to anything, digitalRead() can return either HIGH or LOW (and this can change randomly).

The analog input pins can be used as digital pins, referred to as A0, A1, etc.

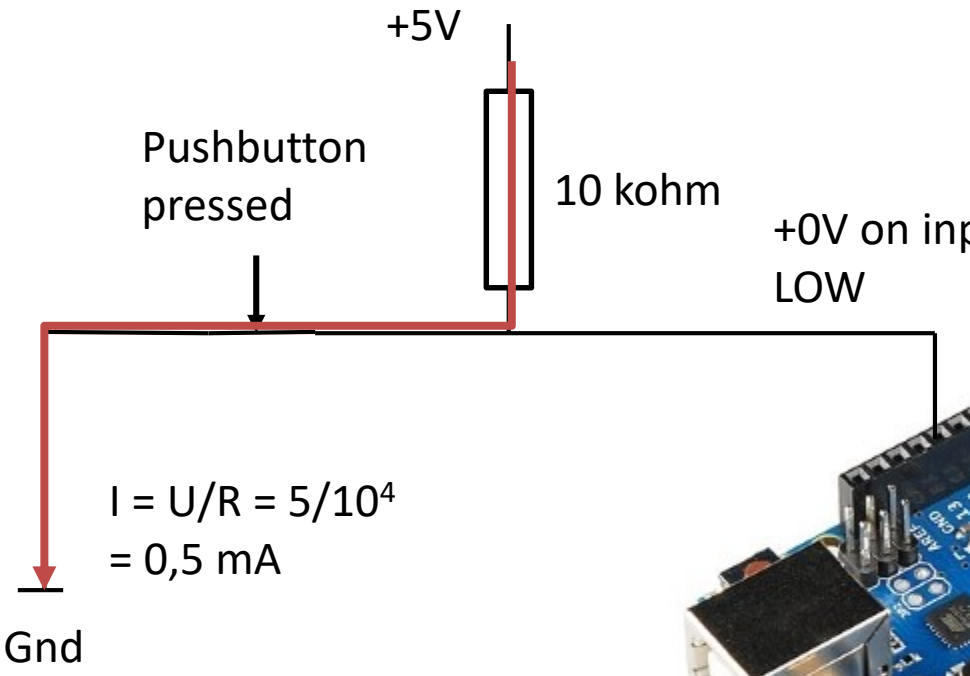


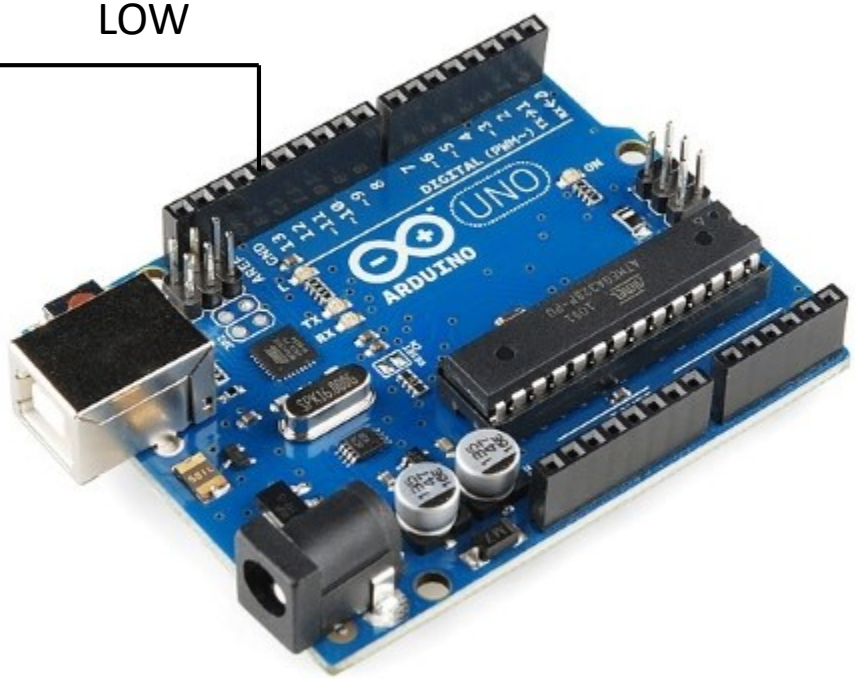
Pull-up resistor.

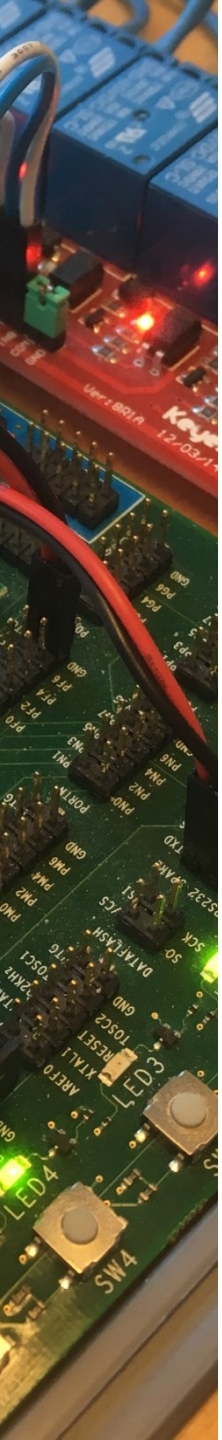




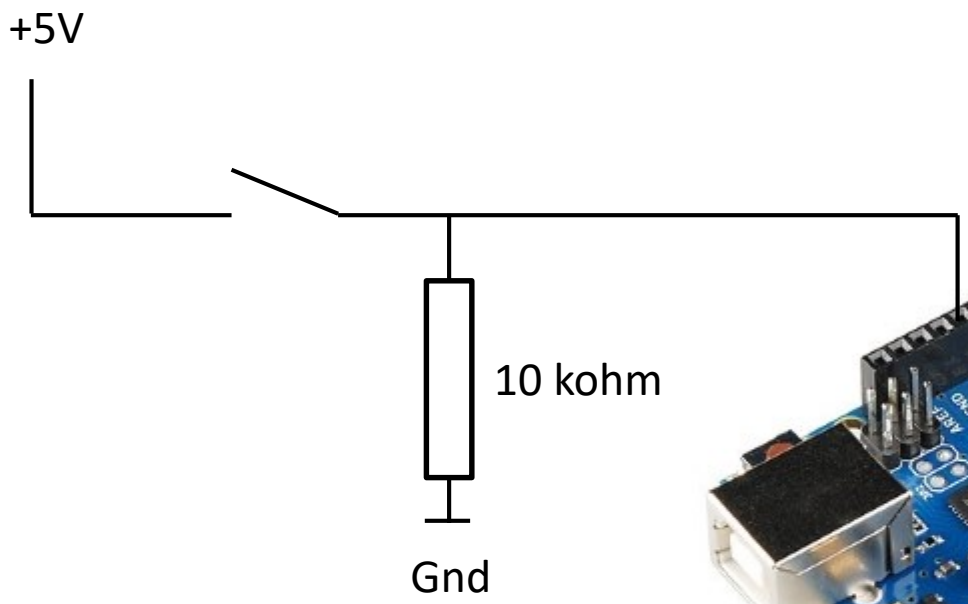
Pull-up resistor.

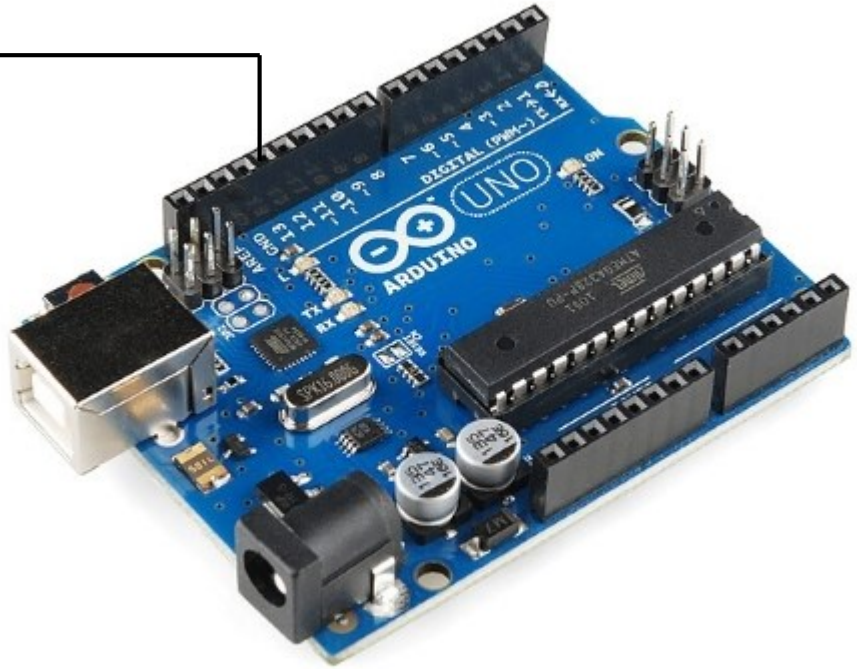


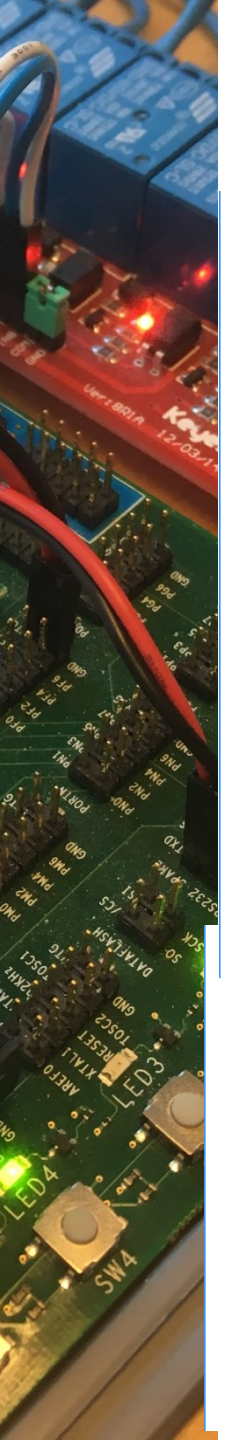




Pull-down resistor.







Arduino, INPUT_PULLUP

Defining Digital Pins modes: INPUT, INPUT_PULLUP, and OUTPUT

Digital pins can be used as INPUT, INPUT_PULLUP, or OUTPUT. Changing a pin with `pinMode()` changes the electrical behavior of the pin.

Pins Configured as INPUT

Arduino (Atmega) pins configured as INPUT with `pinMode()` are said to be in a high-impedance state. Pins configured as `INPUT` make extremely small demands on the circuit that they are sampling, equivalent to a series resistor of 100 Megohms in front of the pin. This makes them useful for reading a sensor.

If you have your pin configured as an `INPUT`, and are reading a switch, when the switch is in the open state the input pin will be "floating", resulting in unpredictable results. In order to assure a proper reading when the switch is open, a pull-up or pull-down resistor must be used. The purpose of this resistor is to pull the pin to a known state when the switch is open. A 10 K ohm resistor is usually chosen, as it is a low enough value to reliably prevent a floating input, and at the same time a high enough value to not not draw too much current when the switch is closed. See the Digital Read Serial tutorial for more information.

If a pull-down resistor is used, the input pin will be `LOW` when the switch is open and `HIGH` when the switch is closed.

If a pull-up resistor is used, the input pin will be `HIGH` when the switch is open and `LOW` when the switch is closed.

Pins Configured as INPUT_PULLUP

The Atmega microcontroller on the Arduino has internal pull-up resistors (resistors that connect to power internally) that you can access. If you prefer to use these instead of external pull-up resistors, you can use the INPUT_PULLUP argument in `pinMode()`.

Defining Pin Levels: HIGH and LOW

When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.

HIGH

The meaning of `HIGH` (in reference to a pin) is somewhat different depending on whether a pin is set to an `INPUT` or `OUTPUT`. When a pin is configured as an `INPUT` with `pinMode()`, and read with `digitalRead()`, the Arduino (Atmega) will report `HIGH` if:

- a voltage greater than 3 volts is present at the pin (5V boards);
- a voltage greater than 2 volts is present at the pin (3.3V boards);

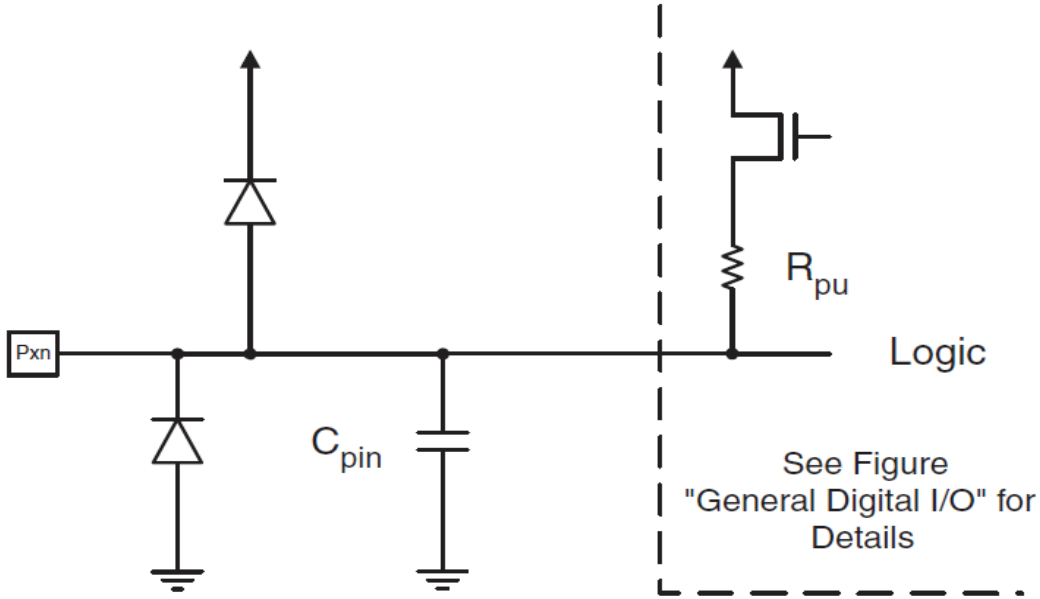
A pin may also be configured as an `INPUT` with `pinMode()`, and subsequently made `HIGH` with `digitalWrite()`. This will enable the internal 20K pullup resistors, which will *pull up* the input pin to a `HIGH` reading unless it is pulled `LOW` by external circuitry. This is how `INPUT_PULLUP` works and is described below in more detail.

doc2549_ATmega2560, page 69:

13.1 Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 13-1. Refer to "Electrical Characteristics" on page 367 for a complete list of parameters.

Figure 13-1. I/O Pin Equivalent Schematic

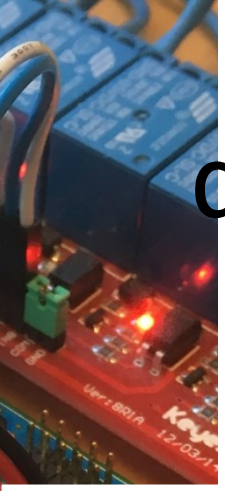


doc2549_ATmega2560, page 70:

All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in “Table 13-34 and Table 13-35 relates the alternate functions of Port L to the overriding signals shown in Figure 13-5 on page 76.” on page 99.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing a logic one to a bit in the PINx Register, will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O” on page 71. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in “Alternate Port Functions” on page 75. Refer to the individual module sections for a full description of the alternate functions.



doc2549_ATmega2560, page 71:

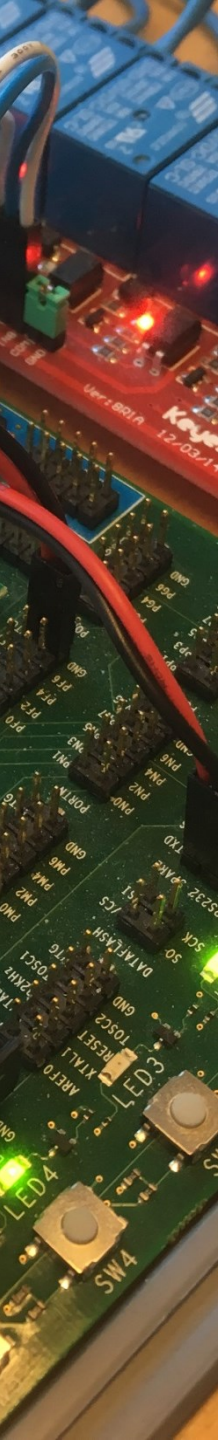
13.2.1 Configuring the Pin

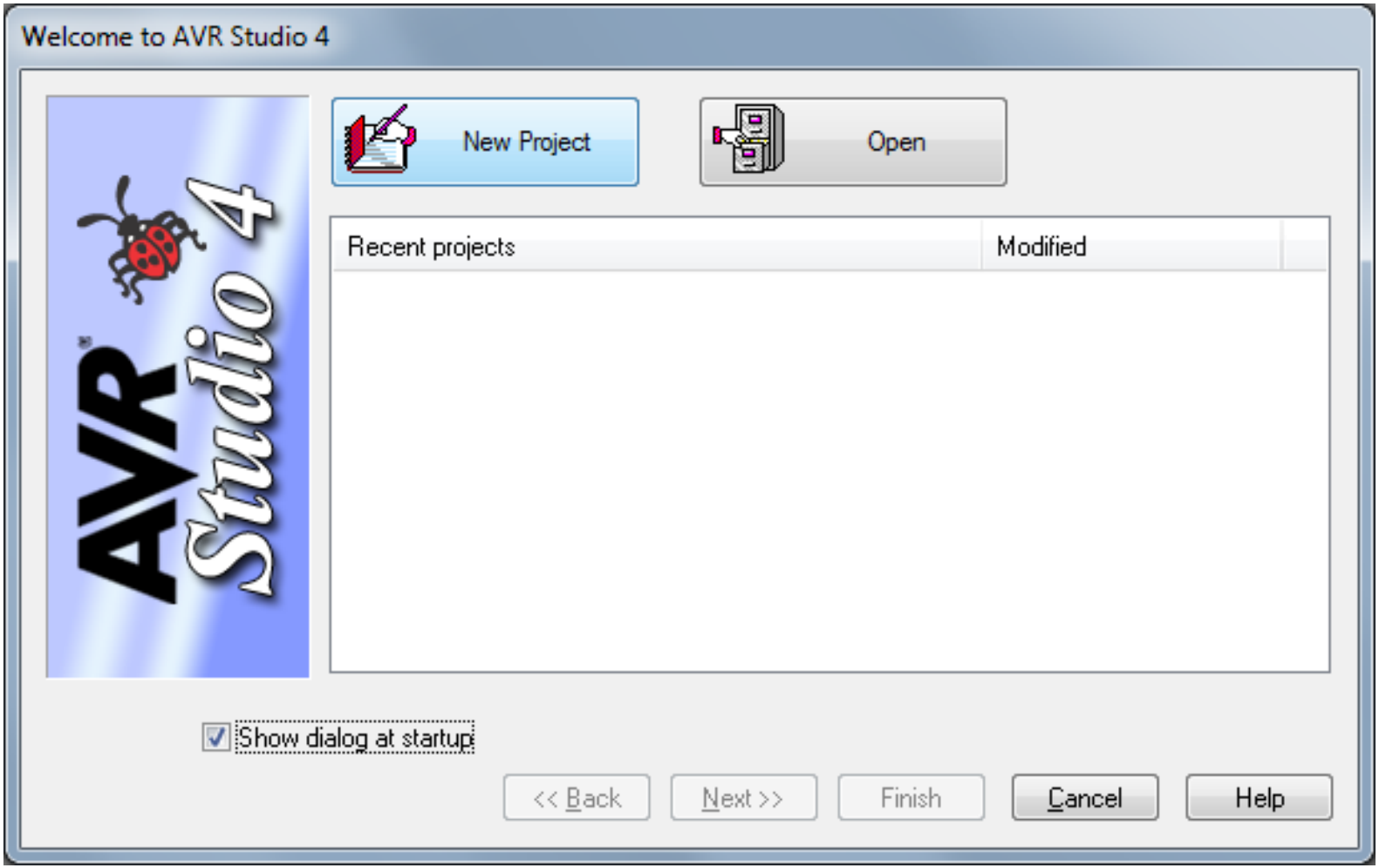
Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Table 13-34 and Table 13-35 relates the alternate functions of Port L to the overriding signals shown in Figure 13-5 on page 76." on page 99, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

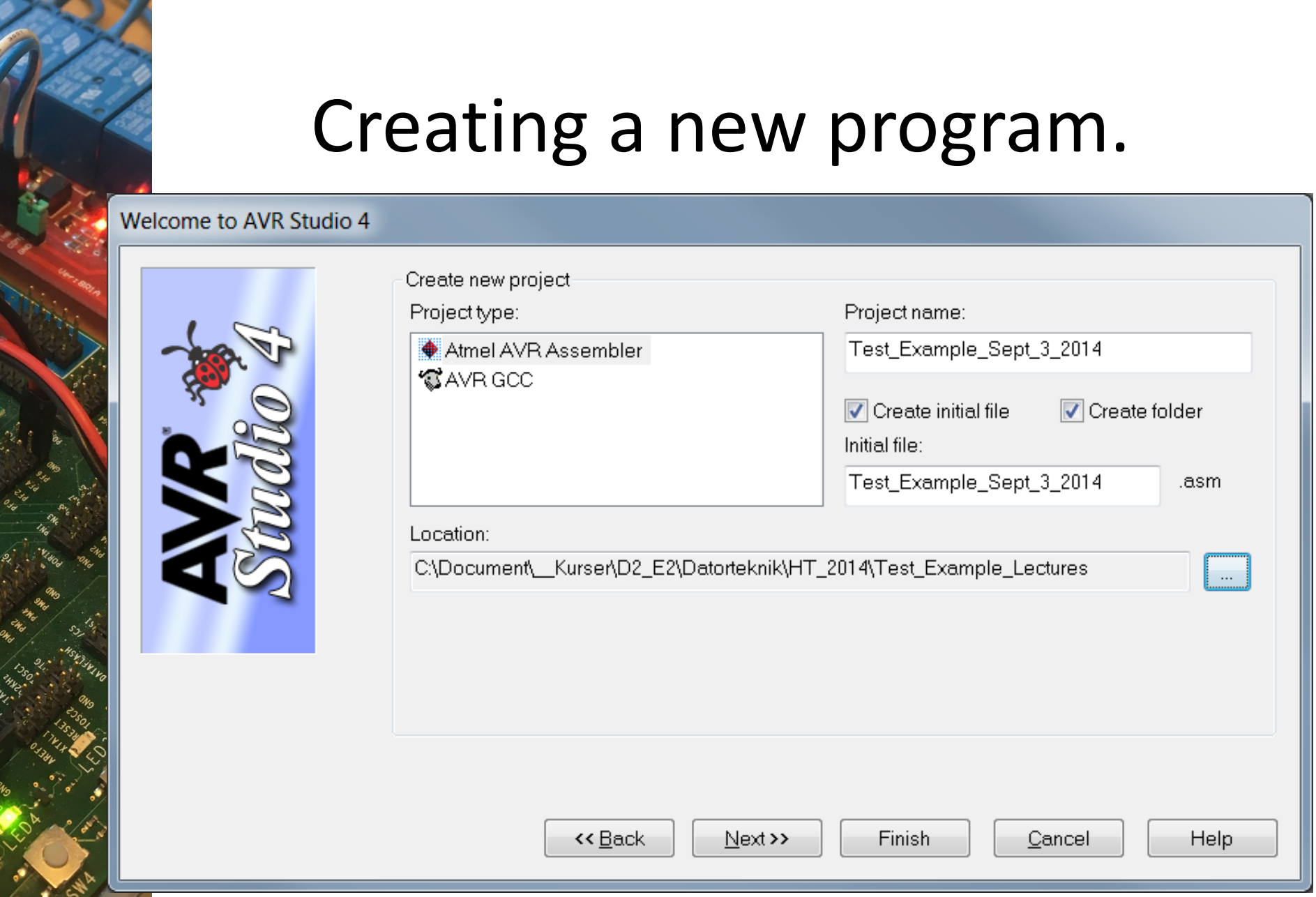
If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

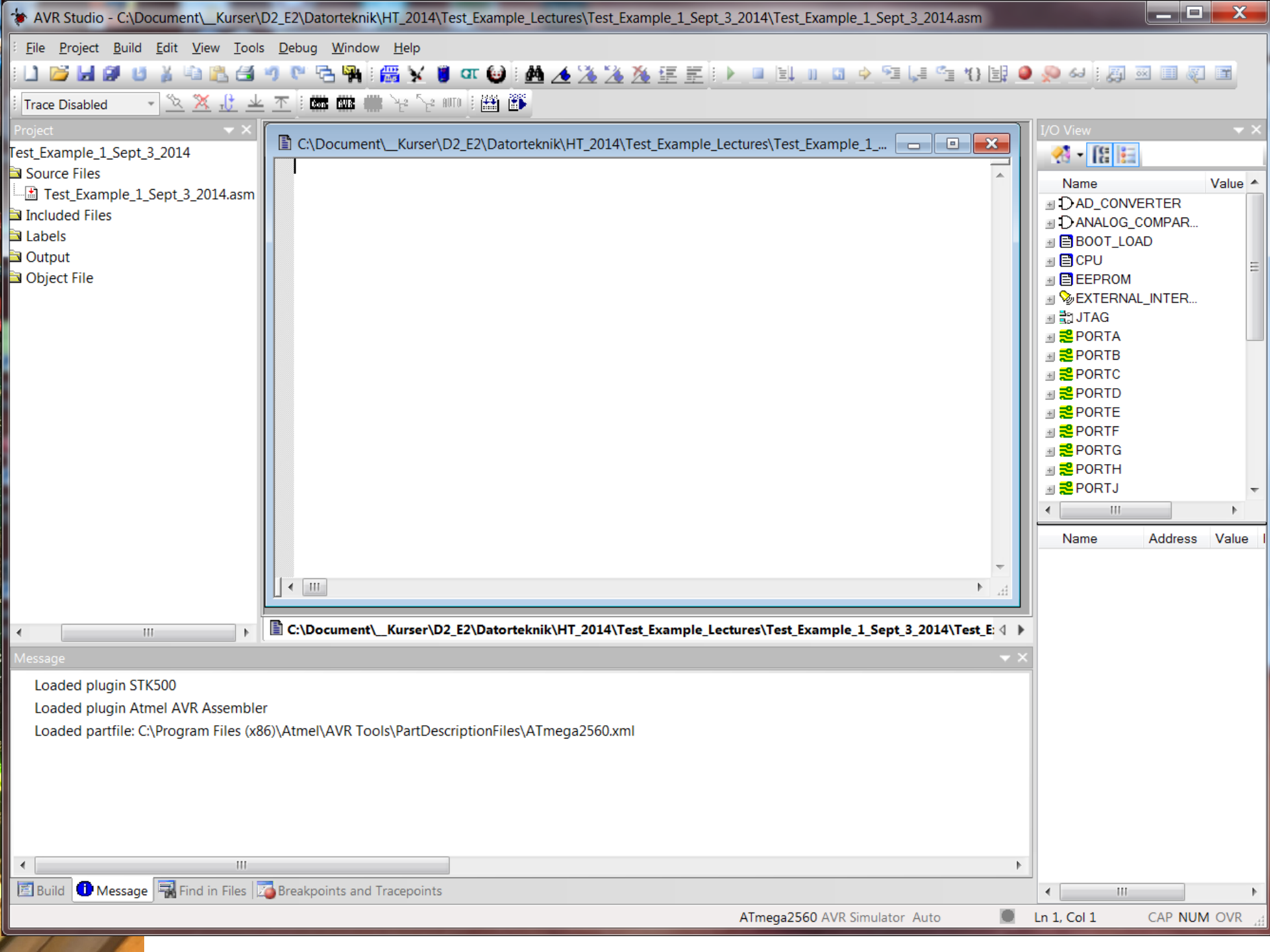
AVR Studio 4

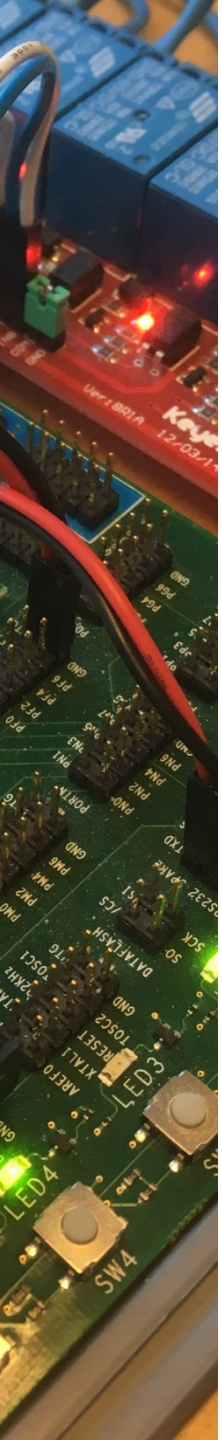



Welcome to AVR Studio 4
AVR Studio 4
New Project
Open
Recent projects
Modified
Show dialog at startup
<< Back
Next >>
Finish
Cancel
Help

Creating a new program.

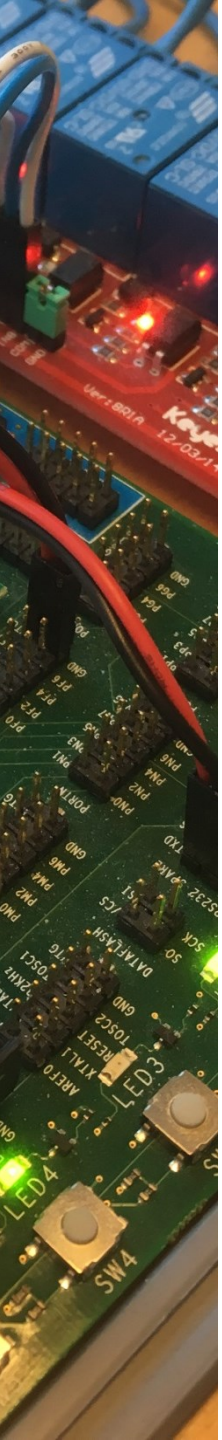



AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
Test_Example_1_Sept_3_2014
Source Files
Test_Example_1_Sept_3_2014.asm
Included Files
Labels
Output
Object File
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_E
I/O View
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
PORTJ
Name
Address
Value
Message
Loaded plugin STK500
Loaded plugin Atmel AVR Assembler
Loaded partfile: C:\Program Files (x86)\Atmel\AVR Tools\PartDescriptionFiles\ATmega2560.xml
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto
Ln 1, Col 1
CAP NUM OVR



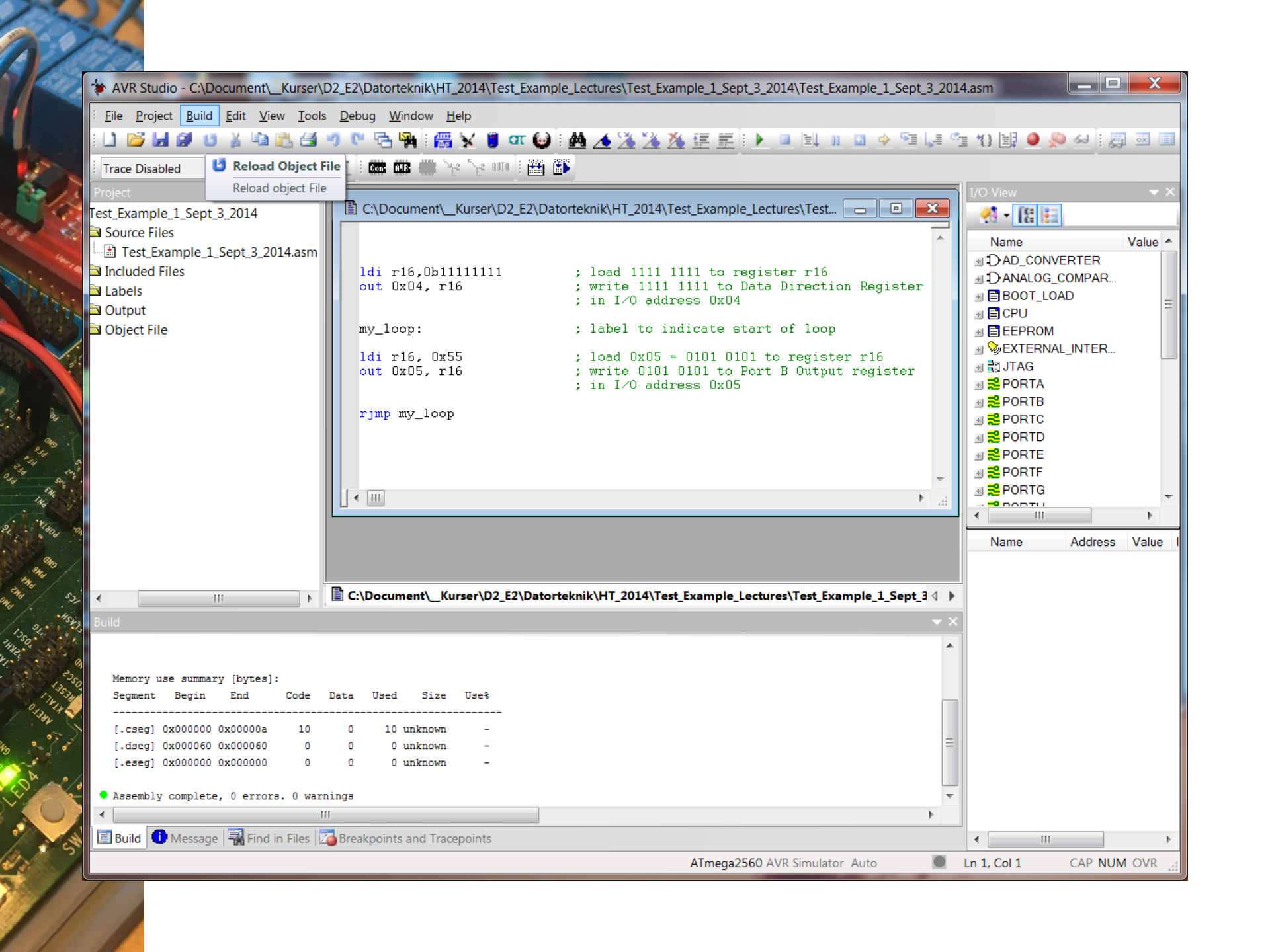
ATmega640/1280/1281/2560/2561

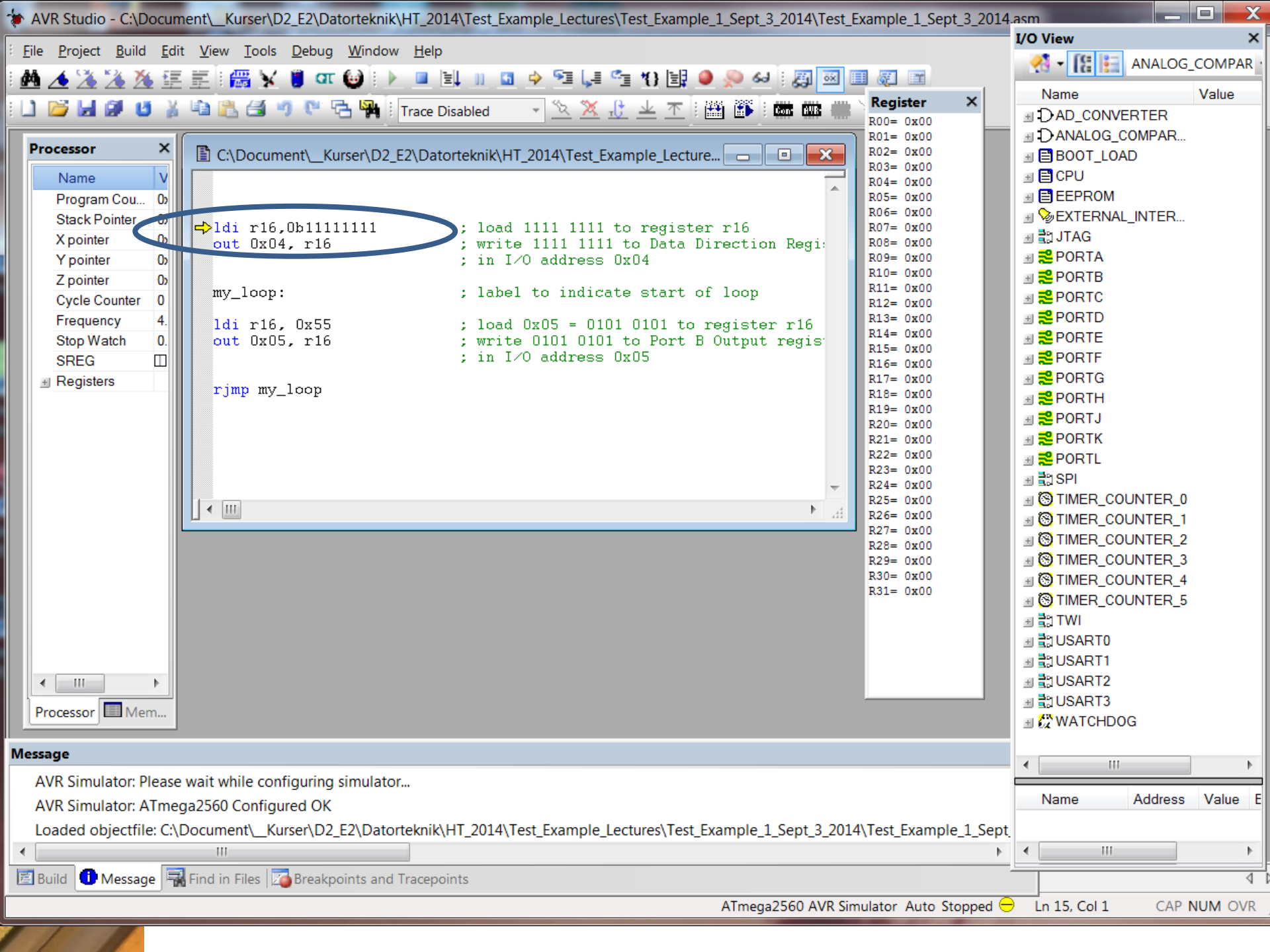
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1A (0x3A)	TIFR5	-	-	ICF5	-	OCF5C	OCF5B	OCF5A	TOV5	166
0x19 (0x39)	TIFR4	-	-	ICF4	-	OCF4C	OCF4B	OCF4A	TOV4	167
0x18 (0x38)	TIFR3	-	-	ICF3	-	OCF3C	OCF3B	OCF3A	TOV3	167
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	193
0x16 (0x36)	TIFR1	-	-	ICF1	-	OCF1C	OCF1B	OCF1A	TOV1	167
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	134
0x14 (0x34)	PORTG	-	-	PORTG5	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0	102
0x13 (0x33)	DDRG	-	-	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	102
0x12 (0x32)	PING	-	-	PING5	PING4	PING3	PING2	PING1	PING0	102
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0	101
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0	102
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0	102
0x0E (0x2E)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0	101
0x0D (0x2D)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0	101
0x0C (0x2C)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0	102
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	101
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	101
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	101
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	101
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	101
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	101
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	100
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	100
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	100
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	100
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	100
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	100

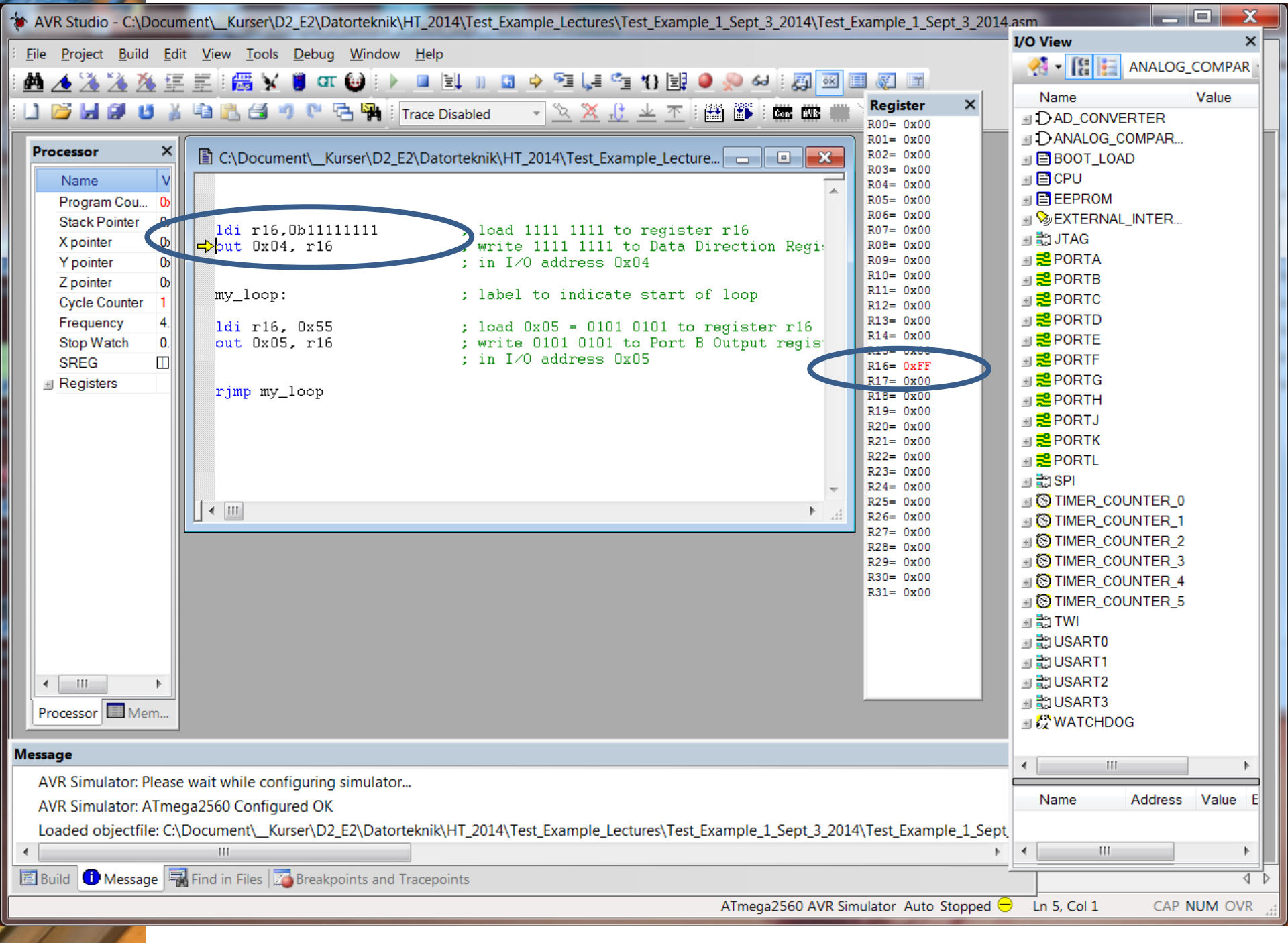


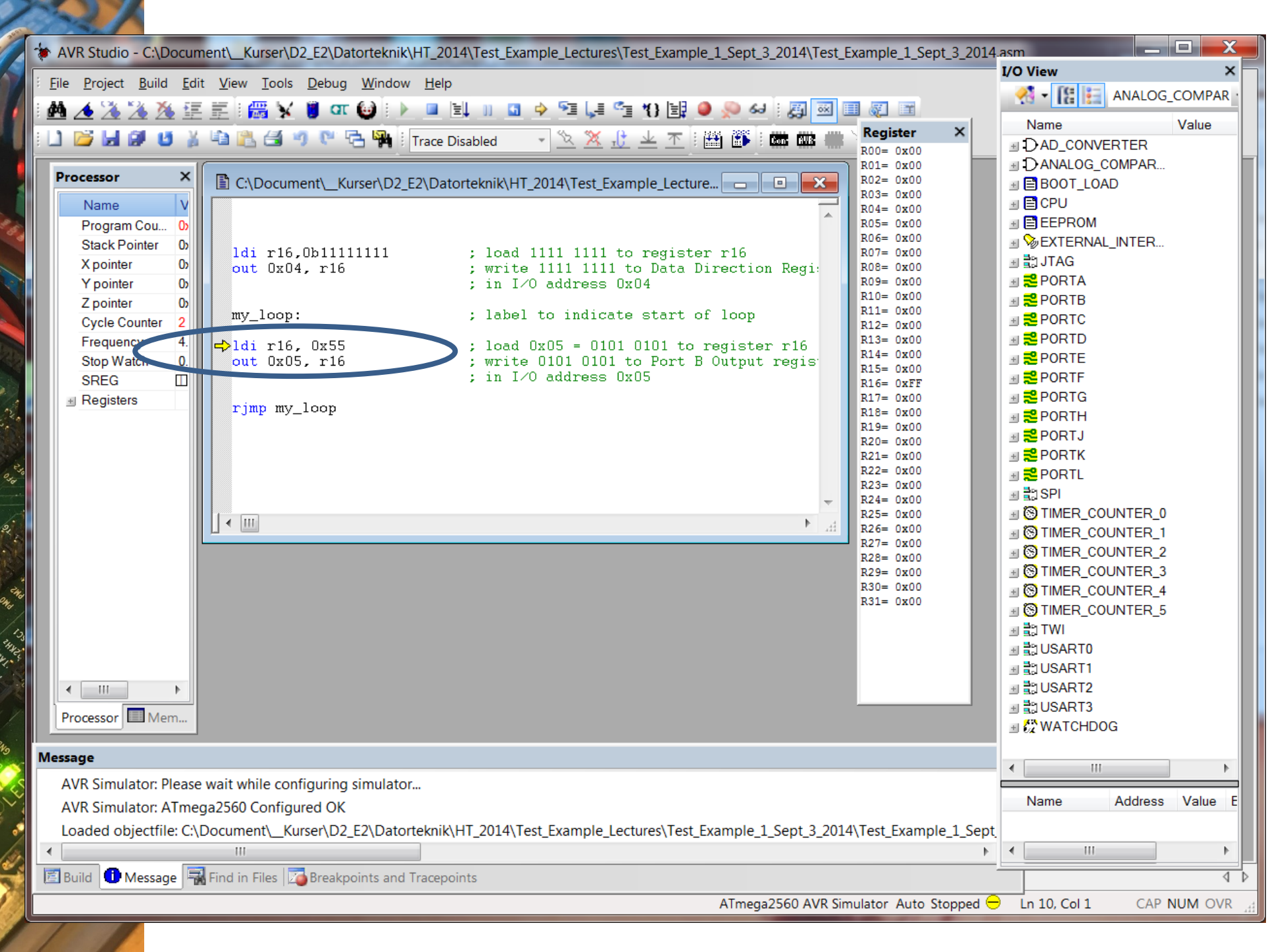
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test...

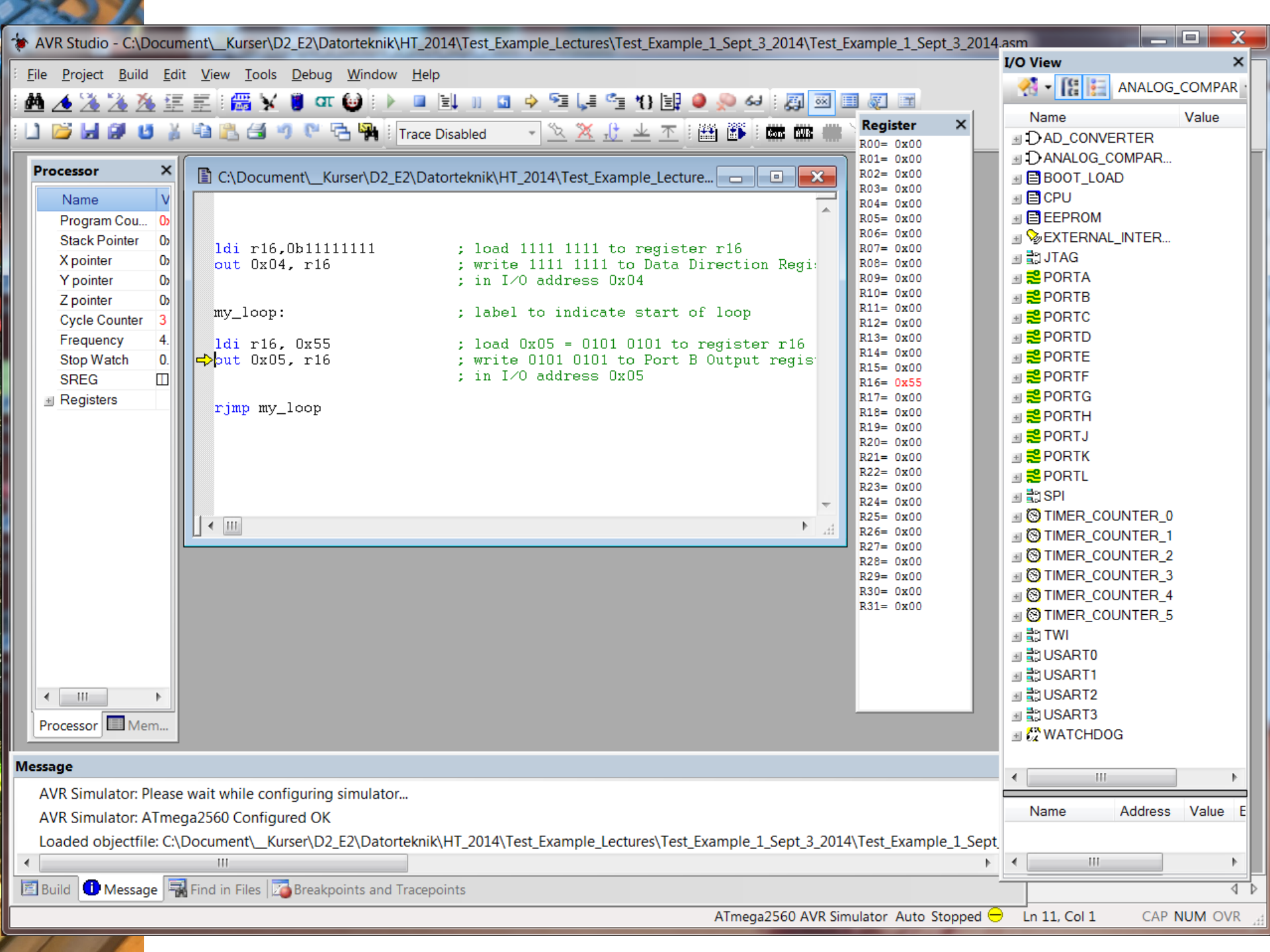
```
ldi r16,0b11111111          ; load 1111 1111 to register r16
out 0x04, r16               ; write 1111 1111 to Data Direction Register
                            ; in I/O address 0x04

my_loop:                    ; label to indicate start of loop

ldi r16, 0x55               ; load 0x05 = 0101 0101 to register r16
out 0x05, r16               ; write 0101 0101 to Port B Output register
                            ; in I/O address 0x05

rjmp my_loop
```

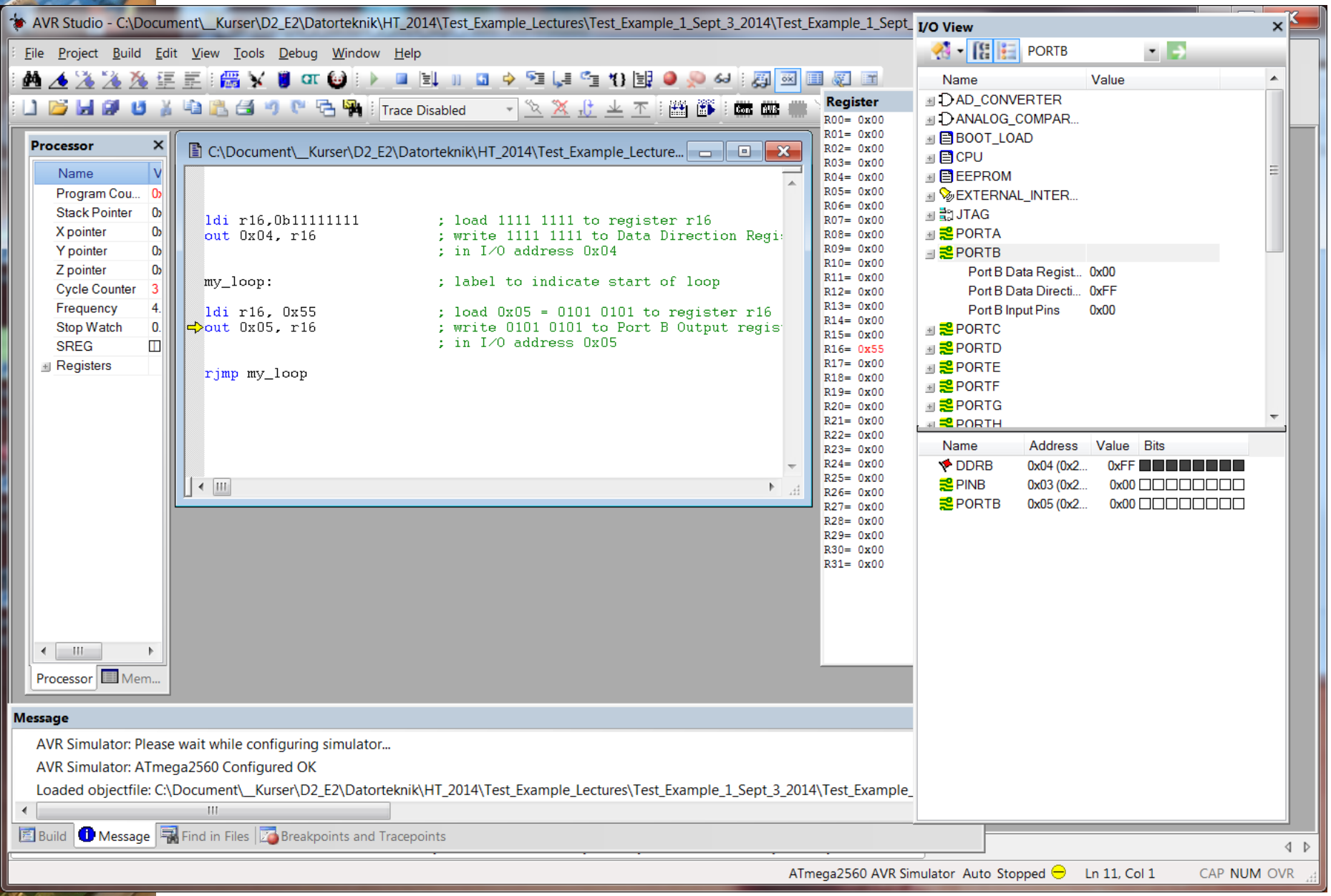

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Reload Object File
Reload object File
Project
Test_Example_1_Sept_3_2014
Source Files
Test_Example_1_Sept_3_2014.asm
Included Files
Labels
Output
Object File
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Register
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output register
; in I/O address 0x05
rjmp my_loop
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3
I/O View
Name Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
Name Address Value
Build
Memory use summary [bytes]:
Segment Begin End Code Data Used Size Use%
[.cseg] 0x000000 0x00000a 10 0 10 unknown -
[.dseg] 0x000060 0x000060 0 0 0 unknown -
[.eseg] 0x000000 0x000000 0 0 0 unknown -
Assembly complete, 0 errors. 0 warnings
Build Message Find in Files Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto
Ln 1, Col 1
CAP NUM OVR

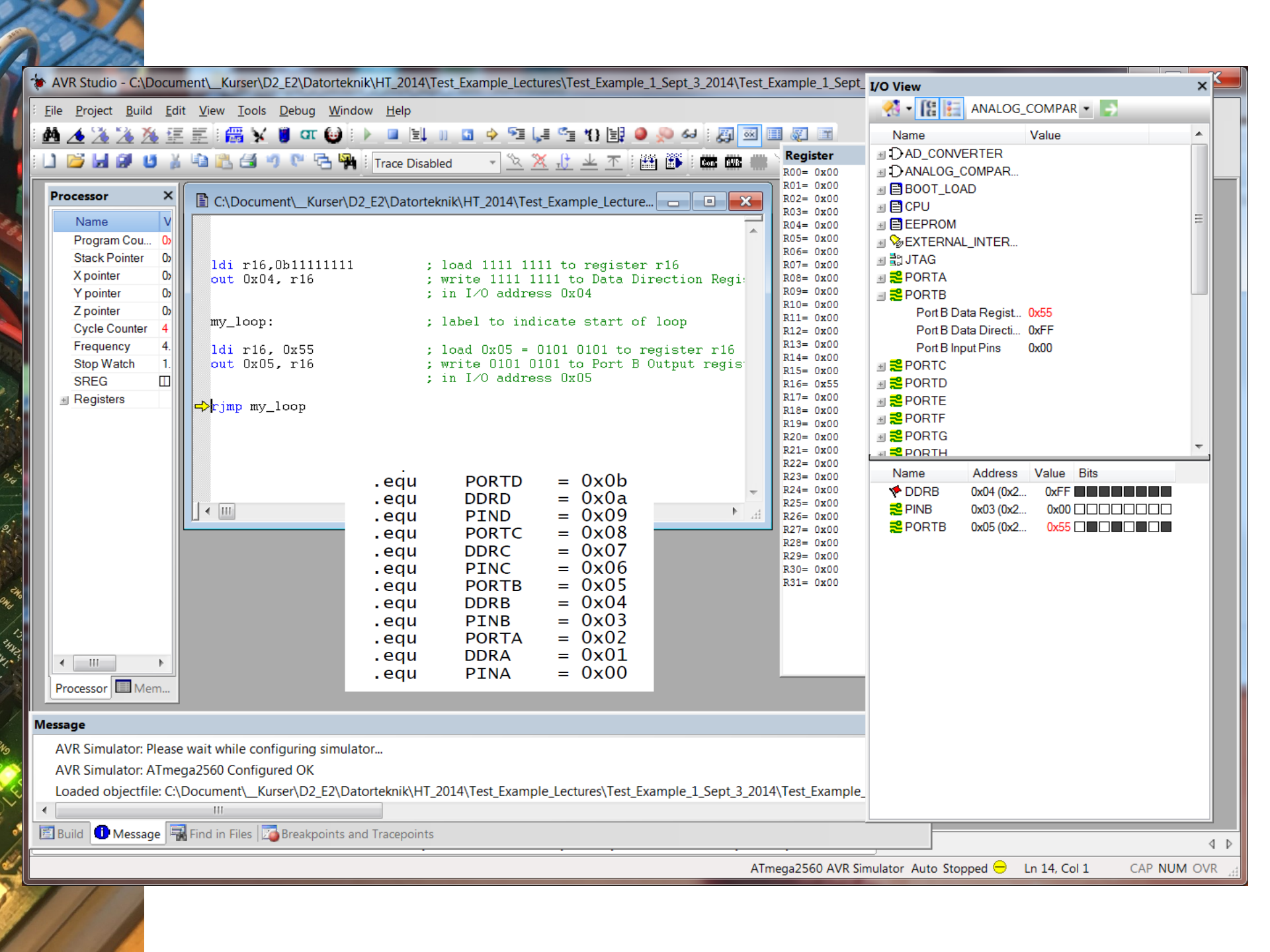

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R15= 0x00
R16= 0x00
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
ANALOG_COMPAR
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
PORTJ
PORTK
PORTL
SPI
TIMER_COUNTER_0
TIMER_COUNTER_1
TIMER_COUNTER_2
TIMER_COUNTER_3
TIMER_COUNTER_4
TIMER_COUNTER_5
TWI
USART0
USART1
USART2
USART3
WATCHDOG
Name
Address
Value
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 15, Col 1
CAP NUM OVR

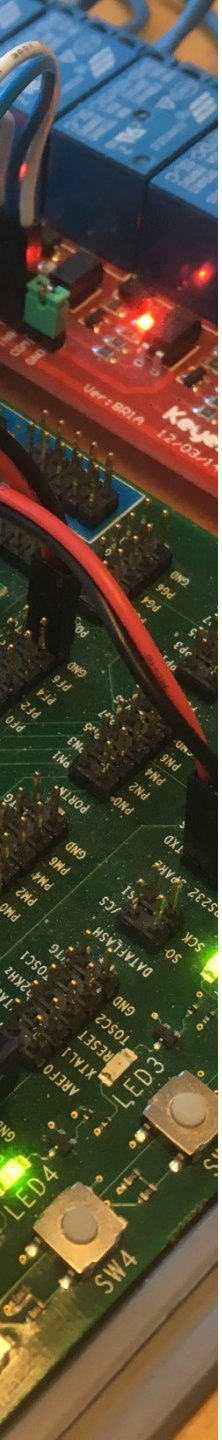

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R16= 0xFF
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
ANALOG_COMPAR
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
PORTJ
PORTK
PORTL
SPI
TIMER_COUNTER_0
TIMER_COUNTER_1
TIMER_COUNTER_2
TIMER_COUNTER_3
TIMER_COUNTER_4
TIMER_COUNTER_5
TWI
USART0
USART1
USART2
USART3
WATCHDOG
Name
Address
Value
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 5, Col 1
CAP NUM OVR


AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R15= 0x00
R16= 0xFF
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
ANALOG_COMPAR
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
PORTJ
PORTK
PORTL
SPI
TIMER_COUNTER_0
TIMER_COUNTER_1
TIMER_COUNTER_2
TIMER_COUNTER_3
TIMER_COUNTER_4
TIMER_COUNTER_5
TWI
USART0
USART1
USART2
USART3
WATCHDOG
Name
Address
Value
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 10, Col 1
CAP NUM OVR


AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R15= 0x00
R16= 0x55
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
ANALOG_COMPAR
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
PORTJ
PORTK
PORTL
SPI
TIMER_COUNTER_0
TIMER_COUNTER_1
TIMER_COUNTER_2
TIMER_COUNTER_3
TIMER_COUNTER_4
TIMER_COUNTER_5
TWI
USART0
USART1
USART2
USART3
WATCHDOG
Name
Address
Value
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 11, Col 1
CAP NUM OVR


AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R15= 0x00
R16= 0x55
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
PORTB
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
Port B Data Regist... 0x00
Port B Data Directi... 0xFF
Port B Input Pins 0x00
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
Name
Address
Value
Bits
DDRB 0x04 (0x2... 0xFF
PINB 0x03 (0x2... 0x00
PORTB 0x05 (0x2... 0x00
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 11, Col 1
CAP NUM OVR


AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
Stop Watch
SREG
Registers
Processor
Mem...
C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lecture...
ldi r16,0b11111111 ; load 1111 1111 to register r16
out 0x04, r16 ; write 1111 1111 to Data Direction Regi
; in I/O address 0x04
my_loop: ; label to indicate start of loop
ldi r16, 0x55 ; load 0x05 = 0101 0101 to register r16
out 0x05, r16 ; write 0101 0101 to Port B Output regis
; in I/O address 0x05
rjmp my_loop
.equ PORTD = 0x0b
.equ DDRD = 0x0a
.equ PIND = 0x09
.equ PORTC = 0x08
.equ DDRC = 0x07
.equ PINC = 0x06
.equ PORTB = 0x05
.equ DDRB = 0x04
.equ PINB = 0x03
.equ PORTA = 0x02
.equ DDRA = 0x01
.equ PINA = 0x00
Register
R00= 0x00
R01= 0x00
R02= 0x00
R03= 0x00
R04= 0x00
R05= 0x00
R06= 0x00
R07= 0x00
R08= 0x00
R09= 0x00
R10= 0x00
R11= 0x00
R12= 0x00
R13= 0x00
R14= 0x00
R15= 0x00
R16= 0x55
R17= 0x00
R18= 0x00
R19= 0x00
R20= 0x00
R21= 0x00
R22= 0x00
R23= 0x00
R24= 0x00
R25= 0x00
R26= 0x00
R27= 0x00
R28= 0x00
R29= 0x00
R30= 0x00
R31= 0x00
I/O View
ANALOG_COMPAR
Name
Value
AD_CONVERTER
ANALOG_COMPAR...
BOOT_LOAD
CPU
EEPROM
EXTERNAL_INTER...
JTAG
PORTA
PORTB
Port B Data Regist... 0x55
Port B Data Directi... 0xFF
Port B Input Pins 0x00
PORTC
PORTD
PORTE
PORTF
PORTG
PORTH
Name
Address
Value
Bits
DDRB 0x04 (0x2... 0xFF
PINB 0x03 (0x2... 0x00
PORTB 0x05 (0x2... 0x55
Message
AVR Simulator: Please wait while configuring simulator...
AVR Simulator: ATmega2560 Configured OK
Loaded objectfile: C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_
Build
Message
Find in Files
Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Stopped
Ln 14, Col 1
CAP NUM OVR



STK600 in ISP mode with ATmega2560

Main | Program | Fuses | LockBits | Advanced | HW Settings | HW Info | Auto

Device

Erase Device

Erase device before flash programming

Verify device after programming

Flash

Use Current Simulator/Emulator FLASH Memory

Input HEX File C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Laboratic ...

Program Verify Read

EEPROM

Use Current Simulator/Emulator EEPROM Memory

Input HEX File C:\Document__Kurser\D2_E2\Datorteknik\HT_2013\Test_pro ...

Program Verify Read

ELF Production File Format

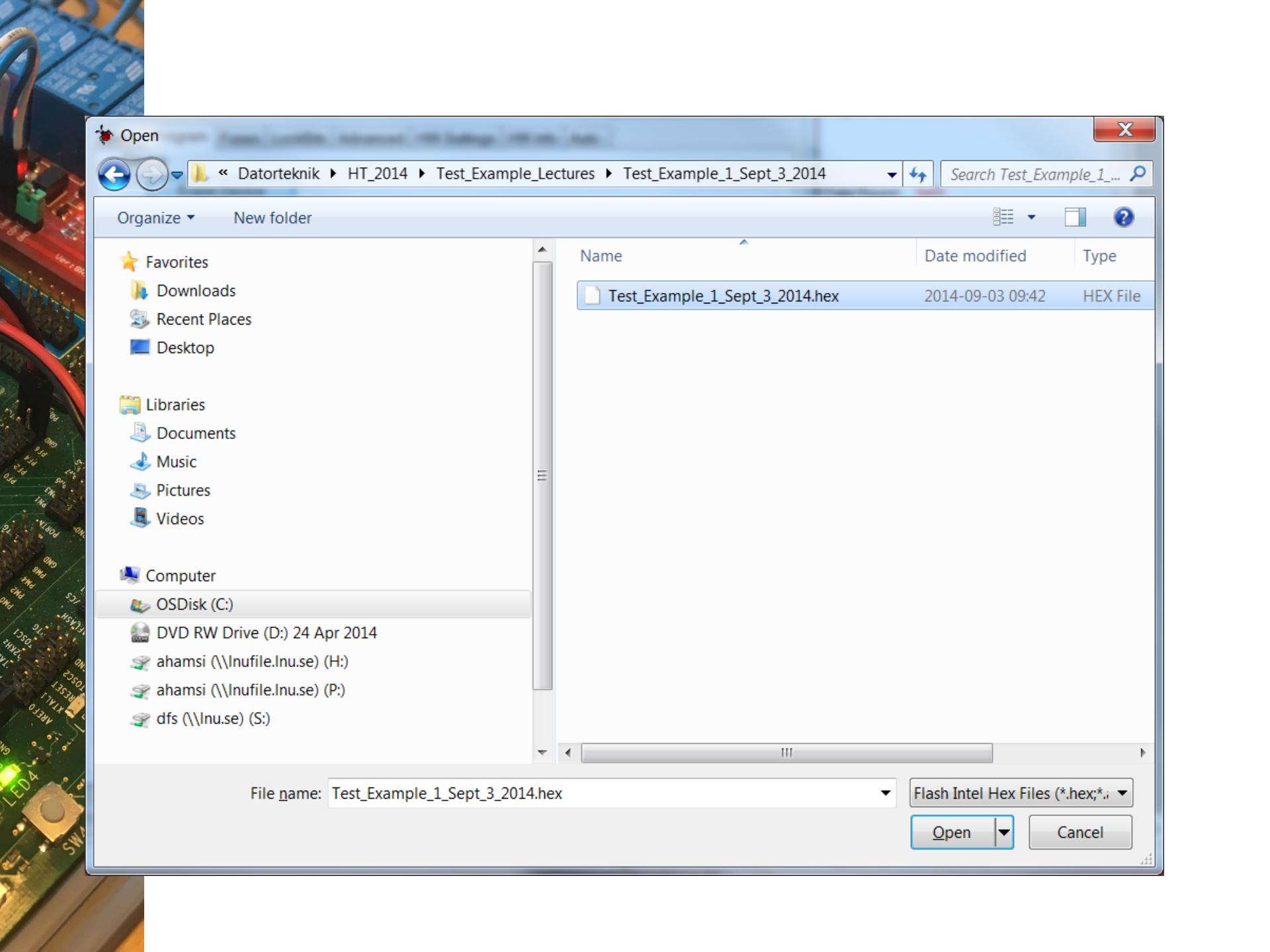
Input ELF File: ...

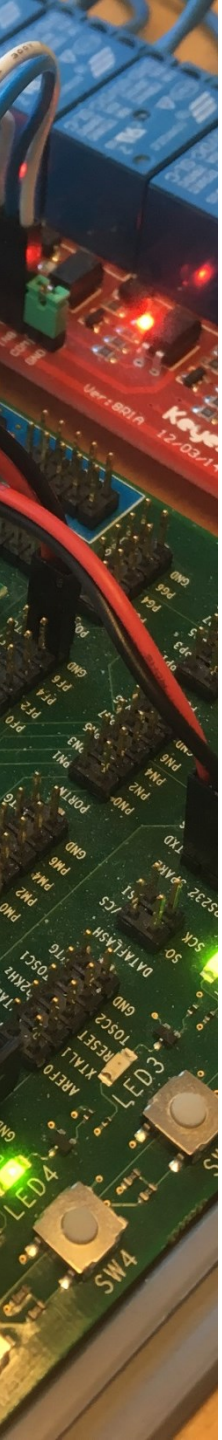
Save From: FLASH EEPROM FUSES LOCKBITS

Fuses and lockbits settings must be specified before saving to ELF

Program Save

Detecting on 'USB'...
STK600 with serial number 004A8D6873A0 found.


Open
« Datorteknik ▸ HT_2014 ▸ Test_Example_Lectures ▸ Test_Example_1_Sept_3_2014
Search Test_Example_1_...
Organize
New folder
Favorites
Downloads
Recent Places
Desktop
Libraries
Documents
Music
Pictures
Videos
Computer
OSDisk (C:)
DVD RW Drive (D:) 24 Apr 2014
ahamsi (\\lnufile.lnu.se) (H:)
ahamsi (\\lnufile.lnu.se) (P:)
dfs (\\lnu.se) (S:)
Name
Date modified
Type
Test_Example_1_Sept_3_2014.hex
2014-09-03 09:42
HEX File
File name:
Test_Example_1_Sept_3_2014.hex
Flash Intel Hex Files (*.hex;*.
Open
Cancel



STK600 in ISP mode with ATmega2560

Main | Program | Fuses | LockBits | Advanced | HW Settings | HW Info | Auto

Device

Erase Device

Erase device before flash programming

Verify device after programming

Flash

Use Current Simulator/Emulator FLASH Memory

Input HEX File .Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.hex ...

Program | Verify | Read

EEPROM

Use Current Simulator/Emulator EEPROM Memory

Input HEX File C:\Document__Kurser\D2_E2\Datorteknik\HT_2013\Test_pro ...

Program | Verify | Read

ELF Production File Format

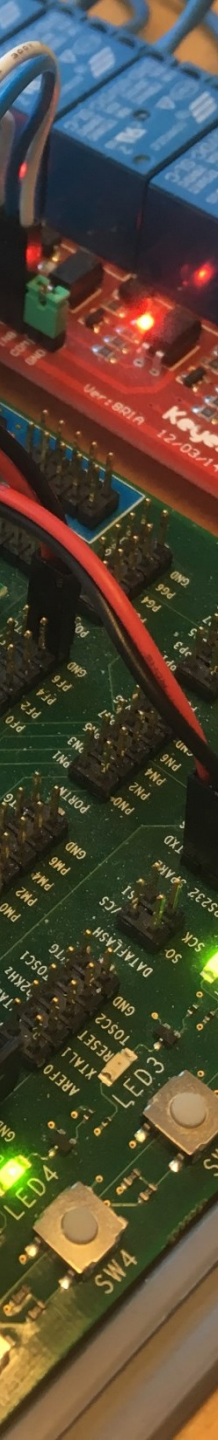
Input ELF File: ...

Save From: FLASH EEPROM FUSES LOCKBITS

Fuses and lockbits settings must be specified before saving to ELF

Program | Save

Programming FLASH .. OK!
Reading FLASH .. OK!
FLASH contents is equal to file.. OK
Leaving programming mode.. OK!



C:\Document__Kurser\D2_E2\Datorteknik\HT_2015\Program_examples\...

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DTr01, Computer Technology I
;   Date: 2015-09-03
;   Author:
;       Anders Haggren
;   Function:
;       Set output
;
;       Example #2
;       Lecture example 2015-09-03, lecture #2
;       Relay card connected to PORTD
;       LEDs connected to PORTB
;       Switches connected to POPTA
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


; Set Data Direction Registers
ldi r16, 0xFF
out 0x04, r16           ; DDRB = 0x04
out 0x0a, r16           ; DDRd = 0x0a

; output registers, PORTB and PORTD

ldi r16, 0x07
out 0x05, r16           ; PORTB = 0x05
out 0x0b, r16           ; PORTd = 0x0b

loop:
rjmp loop
```

Linnéuniversitetet

1DT301, Computer Technology I, autumn 2015
Lab. 1: How to use the PORTs. Digital input/output. Subroutine call.

Goal for this lab:
Learn how to write programs in Assembly language, compile them, download to the STK600 and run them on the target CPU and also to use the AVR Studio 4 Simulator.

Development environment: **AVR Studio4**

The laboratory work has to be done in groups of maximum 2 students per group.

Presentation of results:

Present each task for the teacher when you have solved the task. A written report of all assignments should be submitted after each lab, containing the code and a brief description of results. The report must also include flowcharts for all programs.
The report should be sent to the lab teacher within 1 week, thus before next week lab.
Use text in the programs (comments) to explain the function. Each program must also have a header like the example below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;               1DT301, Computer Technology I
;               Date: 2015-09-03
;               Author:
;                               Student name 1
;                               Student name 2
```

Use text in the programs (comments) to explain the function. Each program must also have a header like the example below.

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;           1DT301, Computer Technology I
;           Date: 2015-09-03
;           Author:
;                       Student name 1
;                       Student name 2
;
;           Lab number: 1
;           Title:      How to use the PORTs. Digital input/output. Subroutine call.
;
;           Hardware:   STK600, CPU ATmega2560
;
;           Function: Describe the function of the program, so that you can understand it,
;                     even if you're viewing this in a year from now!
;
;           Input ports: Describe the function of used ports, for example on-board switches
;                       connected to PORTA.
;
;           Output ports: Describe the function of used ports, for example on-board LEDs
;                       connected to PORTB.
;
;           Subroutines:            If applicable.
;           Included files:         m2560def.inc
;
;           Other information:
;
;           Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Task 1:

Write a program in Assembly language to light LED 2.
You can use any of the four ports, but start with PORTB.
The program should be very short! How many instructions is minimum number?

Task 2:

Write a program in Assembly language to read the switches and light the corresponding LED.
Example: When you press SW5, LED5 so should light.

Make an initialization part of the program and after that an infinite loop.

Task 3:

Write a program in Assembly language to read the switches and light LED0 then you press SW5.
For all other switches there should be no activity.

Task 4:

Run the program in Task 3 in the simulator.

Task 5:

Task 5:

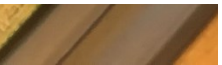
Write a program in Assembly language that creates a Ring Counter. The values should be displayed with the LEDs. Use shift instructions, LSL or LSR.
Make a delay of approximately 0.5 sec in between each count. Write the delay as a subroutine. For using the subroutine, you must initialize the Stack Pointer, SP. Include the following instructions in beginning of your program:

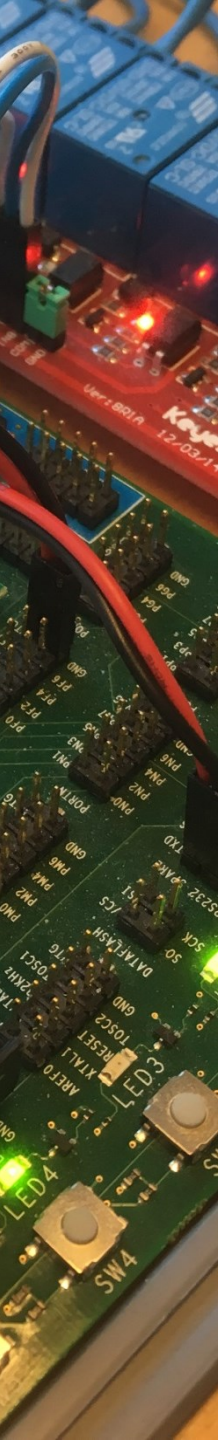
```
; Initialize SP, Stack Pointer
ldi r20, HIGH(RAMEND)        ; R20 = high part of RAMEND address
out SPH,R20                  ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)         ; R20 = low part of RAMEND address
out SPL,R20                  ; SPL = low part of RAMEND address
```

Function, the 8 LEDs:
(0000 000X, 0000 00X0, 0000 0X00, 0000 X000, 000X 0000, 00X0 0000, 0X00 0000, X000 0000)

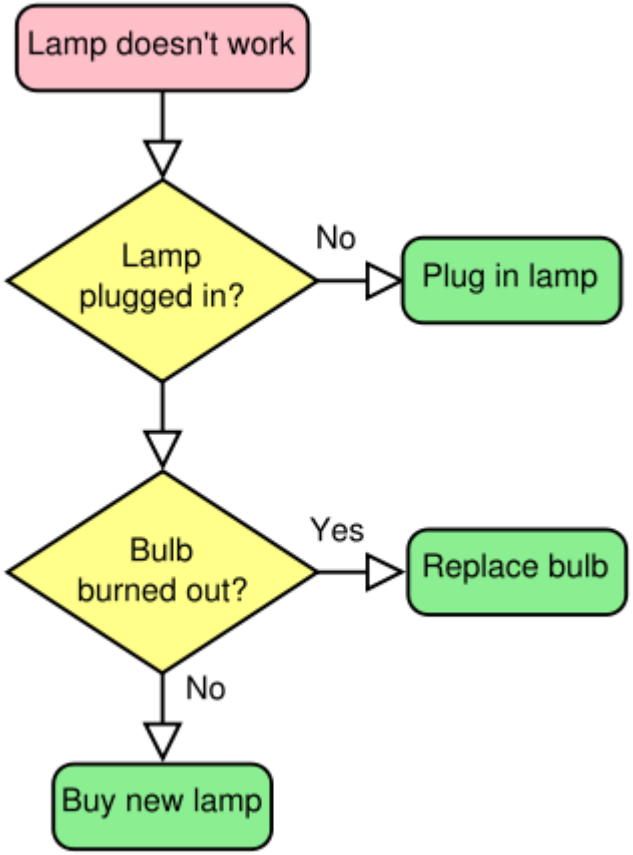
Task 6:
Write a program in Assembly language that creates a Johnson Counter in an infinite loop.

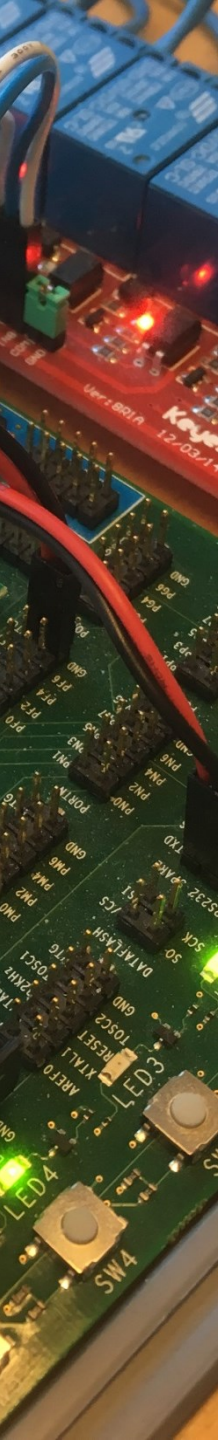
Function, the 8 LEDs:
(0000 000X, 0000 00XX, 0000 0XXX, 0000 XXXX, 000X XXXX, 00XX XXXX, 0XXX XXXX, XXXX XXXX, 0XXX XXXX, 00XX XXXX, 000X XXXX, 0000 XXXX, 0000 0XXX, 0000 00XX, 0000 000X, 0000 0000)



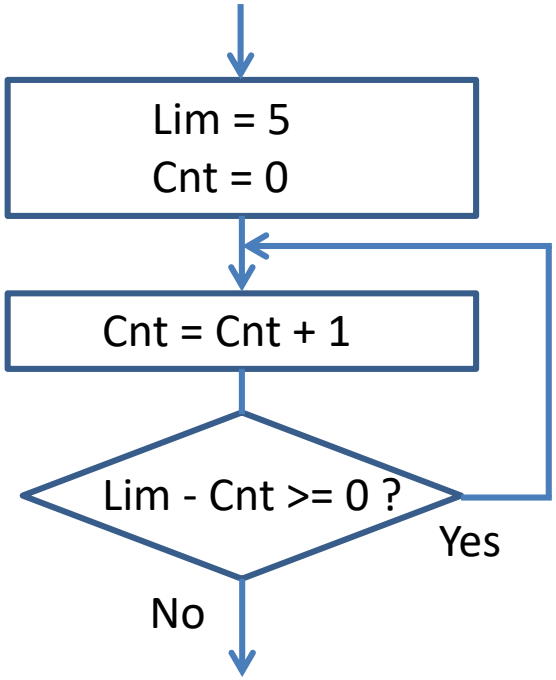


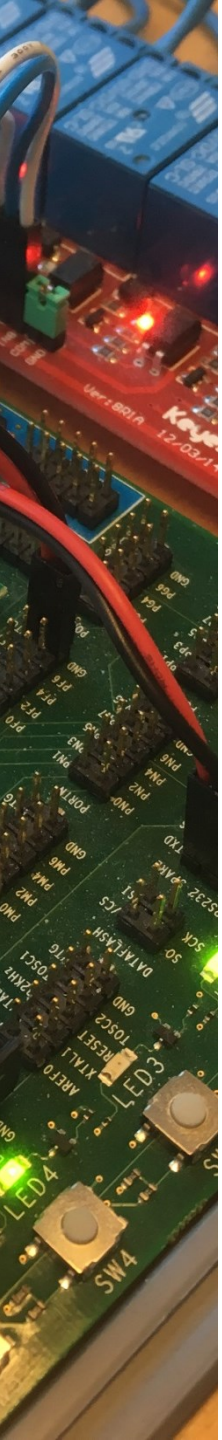
Using Flow Charts





Loop:

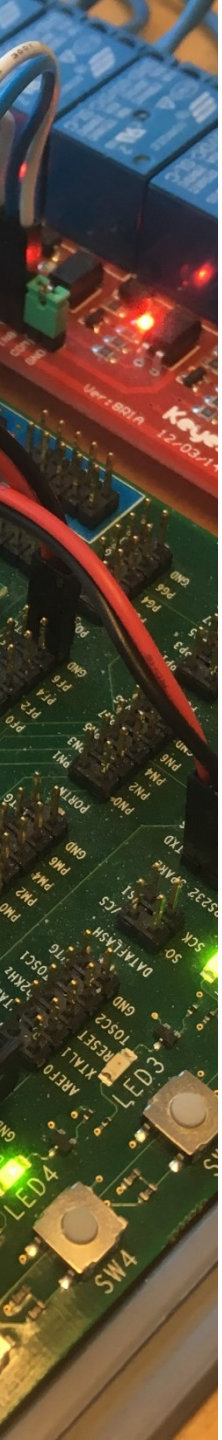




Simple delay loop

C:\Document__Kurser\D2_E2\Datorteknik\HT_2015\Program_...

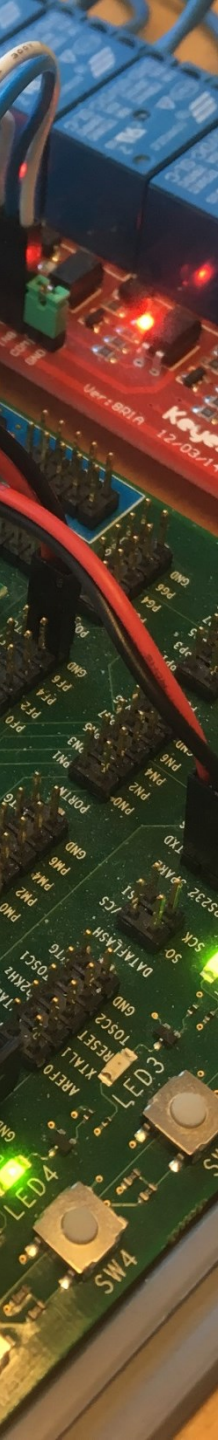
```
Delay:

    ldi r16, 5   ; r16 = limit value
    ldi r17, 0   ; r17 = loop counter

del_1:
    inc r17      ; r17 = r17 + 1
    cp r16, r17  ; compare r16 - 17
    brge del_1   ; if greater or equal, go back

rjmp my_loop
```



Program example

C:\Document__Kurser\D2_E2\Datorteknik\HT_2015\Program_...

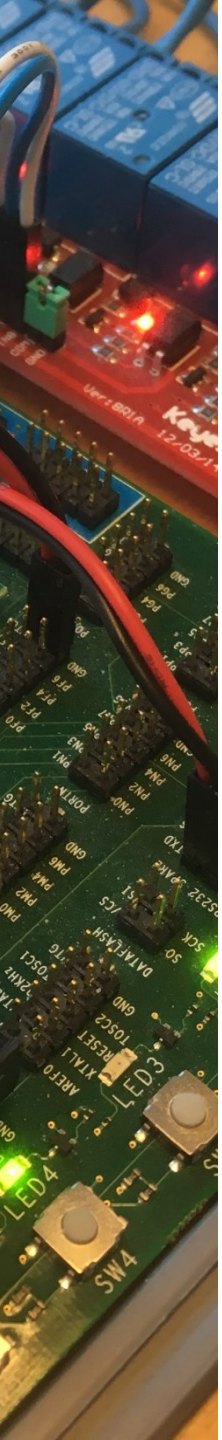
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2015-09-03
;   Author:
;       Anders Haggren
;   Function:
;       Set output
;
;       Example #2
;       Lecture example 2015-09-03, lecture #2
;       Relay card connected to PORTD
;       LEDs connected to PORTB
;       Switches connected to POPTA
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


; Set Data Direction Registers
ldi r16, 0xFF
out 0x04, r16           ; DDRB = 0x04
out 0x0a, r16           ; DDRD = 0x0a


ldi r17, 0xFF           ; set all bits to 1
ldi r18, 255            ; set all bits to 1
ldi r19, 0b11111111     ; set all bits to 1
ldi r16, -1             ; set all bits to 1

; output registers, PORTB and PORTD
my_loop:

ldi r16, 0
out 0x05, r16           ; PORTB = 0x05
out 0x0b, r16           ; PORTD = 0x0b

rjmp my_loop
```



Initialize Stack pointer, SP

C:\Document__Kurser\D2_E2\Datorteknik\HT_2015\Program_examples\Ex_2_Lecture_2_Se...

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1DT301, Computer Technology I
;    Date: 2015-09-03
;    Author:
;        Anders Haggren
;    Function:
;        Set output
;
;        Example #2
;        Lecture example 2015-09-03, lecture #2
;        Relay card connected to PORTD
;        LEDs connected to PORTB
;        Switches connected to POPTA
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"     ; include file for Atmega 2560
; by including the file m2560def.inc names can be used instead of addresses
; for exampl DDRA, DDRB, PORTA, PORTB, PINA, PINB etc

;Initialize SP, Stack Pointer
;Necessary for using instructions rcall and ret

ldi r20, HIGH(RAMEND)    ; R20 = high part of RAMEND address
out SPH,R20              ; SPH = high part of Stackpointer
ldi R20, low(RAMEND)     ; R20 = low part of RAMEND address
out SPL,R20              ; SPL = low part of Stackpointer


; Set Data Direction Registers
ldi r16, 0xFF
out DDRB, r16            ; DDRB = 0x04
out DDRD, r16            ; DDRD = 0x0a
```



CP – Compare

Description:

This instruction performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - Rr

	Syntax:	Operands:	Program Counter:
(i)	CP Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0001	01rd	dddd	rrrr

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet Rr3 + Rr3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

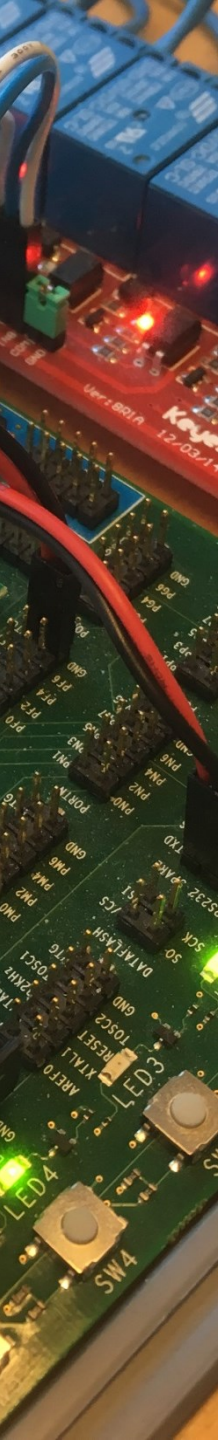
V: $Rd7 \bullet \overline{Rr7} \bullet \overline{R7} + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: $\overline{Rd7} \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.



CPI – Compare with Immediate

Description:

This instruction performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - K

	Syntax:	Operands:	Program Counter:
(i)	CPI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-bit Opcode:

0011	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet K3 + K3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

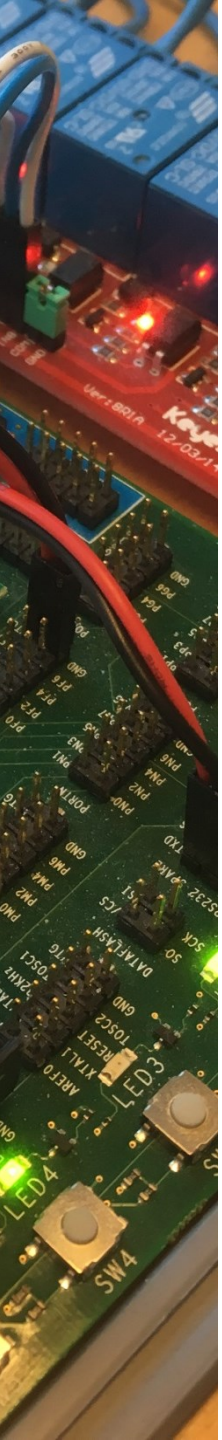
V: $Rd7 \bullet \overline{K7} \bullet \overline{R7} + \overline{Rd7} \bullet K7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: $\overline{Rd7} \bullet K7 + K7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.



DEC – Decrement

Description:

Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd.

The C Flag in SREG is not affected by the operation, thus allowing the DEC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned values, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) Rd ← Rd - 1

	Syntax:	Operands:	Program Counter:
(i)	DEC Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	1010

Status Register and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

S: N ⊕ V
For signed tests.

V: $\overline{R7}$ •R6 •R5 •R4• R3• R2 •R1• R0
Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was $80 before the operation.

N: R7
Set if MSB of the result is set; cleared otherwise.

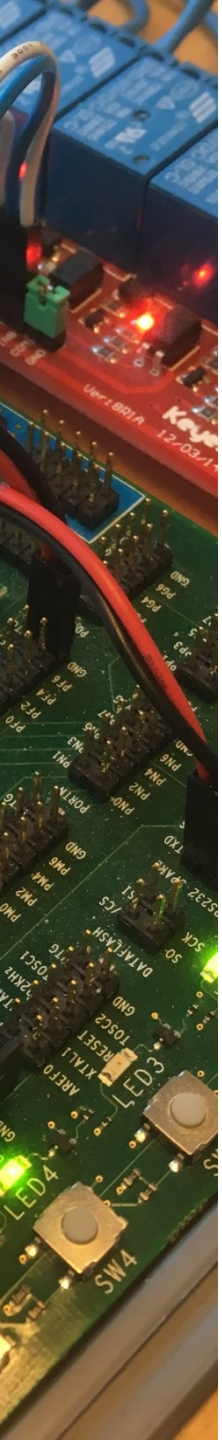
Z: $\overline{R7}$ •$\overline{R6}$• $\overline{R5}$ •$\overline{R4}$• $\overline{R3}$• $\overline{R2}$• $\overline{R1}$• $\overline{R0}$
Set if the result is $00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        ldi   r17,$10   ; Load constant in r17
loop:   add   r1,r2     ; Add r2 to r1
        dec   r17       ; Decrement r17
        brne  loop      ; Branch if r17<>0
        nop             ; Continue (do nothing)
```

Words: 1 (2 bytes)



BRGE – Branch if Greater or Equal (Signed)

Description:

Conditional relative branch. Tests the Signed Flag (S) and branches relatively to PC if S is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was greater than or equal to the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 4,k).

Operation:

(i) If Rd ≥ Rr (N ⊕ V = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRGE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k100

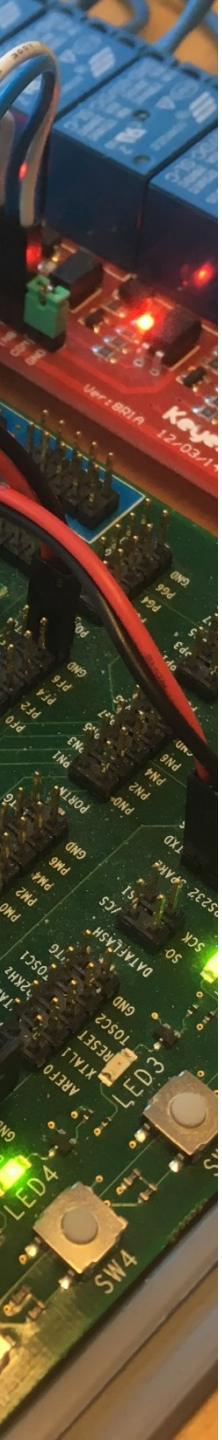
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
            cp    r11,r12    ; Compare registers r11 and r12
            brge  greateq    ; Branch if r11 ≥ r12 (signed)
            ...
greateq:    nop              ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true



BREQ – Branch if Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k).

Operation:

(i) If Rd = Rr (Z = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BREQ k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k001

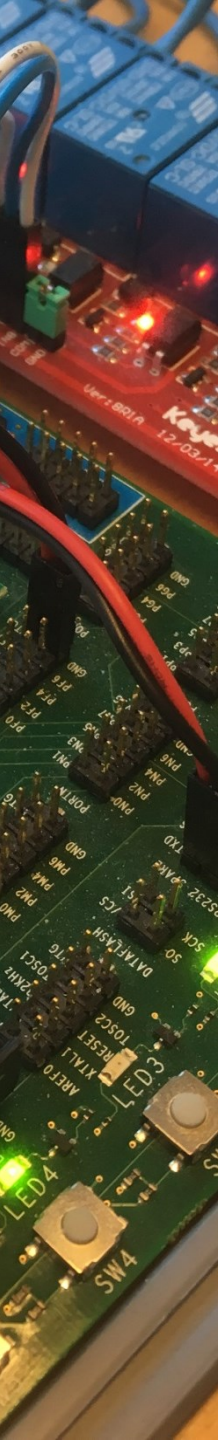
Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
        cp    r1,r0     ; Compare registers r1 and r0
        breq  equal     ; Branch if registers equal
        ...
equal:  nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true



BRNE – Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRNE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k001

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

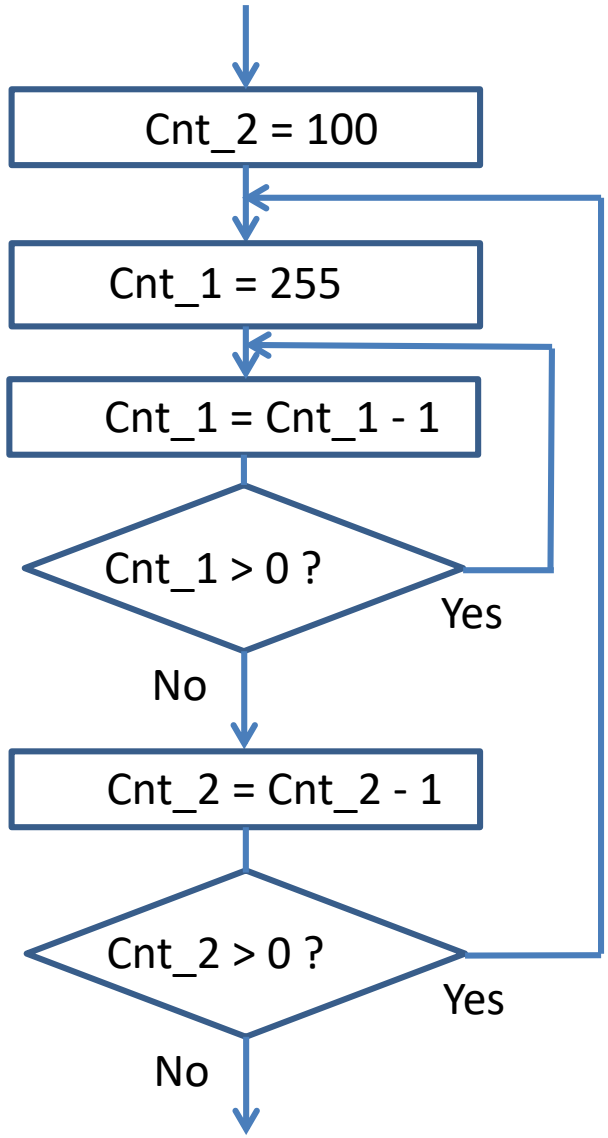
```
          eor    r27,r27   ; Clear r27
loop:     inc    r27       ; Increase r27
          ...
          cpi    r27,5     ; Compare r27 to 5
          brne   loop      ; Branch if r27<>5
          nop              ; Loop exit (do nothing)
```

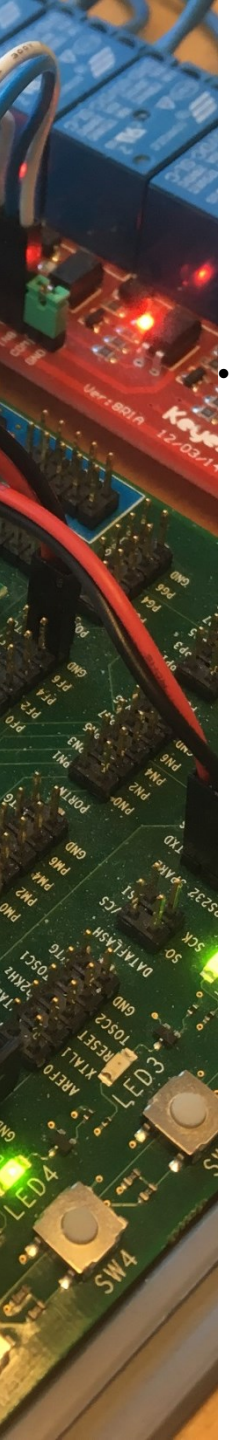
Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true

Example

Example,
double delay loop.

```
delay:      ; Example of double delay loop

    ldi r19,2           ; cnt_2 = 100

out_loop:
    ldi r18,5           ; cnt_1 = 255

in_loop:
    dec r18             ; cnt_1 = cnt_1 -1
    cpi r18, 0
    brne in_loop

    dec r19             ; cnt_2 = cnt_2 -1
    cpi r19,0
    brne out_loop

    ret
```





Run simulator in AVR Studio 4.

- C:\Document__Kurser\D2_E2\Datorteknik\HT_2019\Program_examples\test\Lecture_2_Sept_4_2019