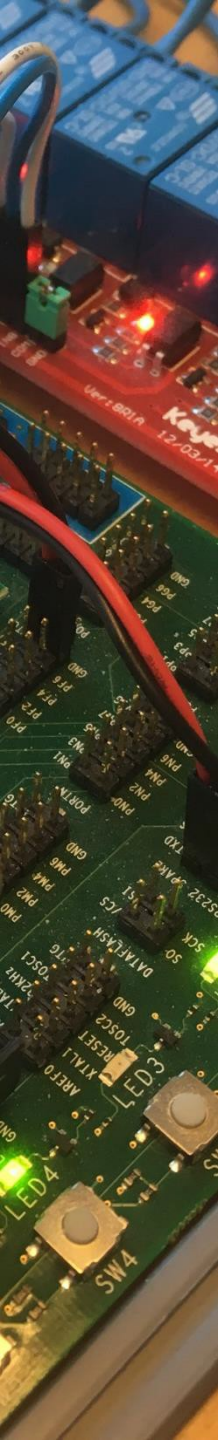
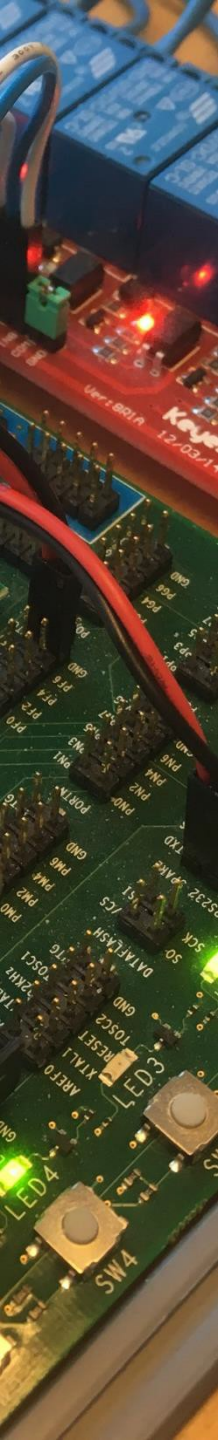
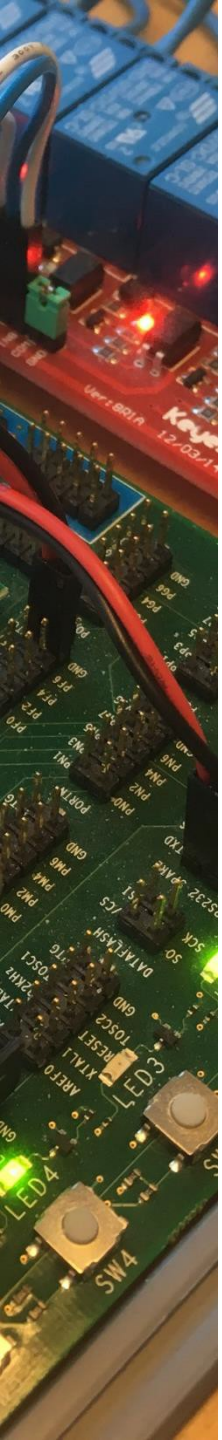


1DT301, Computer Technology
Lecture #2,
Program example,
some instructions,
STK600.
Introduction to Lab 1.

STK 600





Documentation:

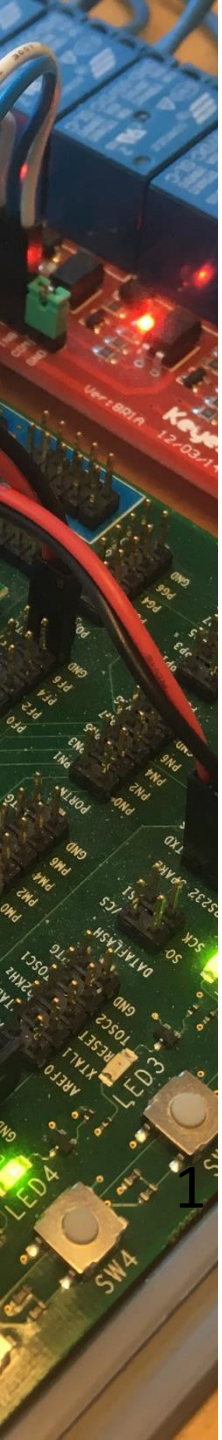
STK600, ATMega2560:

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

STK500, ATMega16:

- doc2466_ATmega16.pdf - ATmega 16, 8-bit AVR Microcontroller (STK500)

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



Binary and hexadecimal numbers.

0001 1110 1101 0000 0010 0001

$0 * 2^{23} + 0 * 2^{22} + 0 * 2^{21} + 1 * 2^{20} + 1 * 2^{19} + 1 * 2^{18} + 1 * 2^{17} + 0 * 2^{16} + \ldots$

1 E D 0 2 1

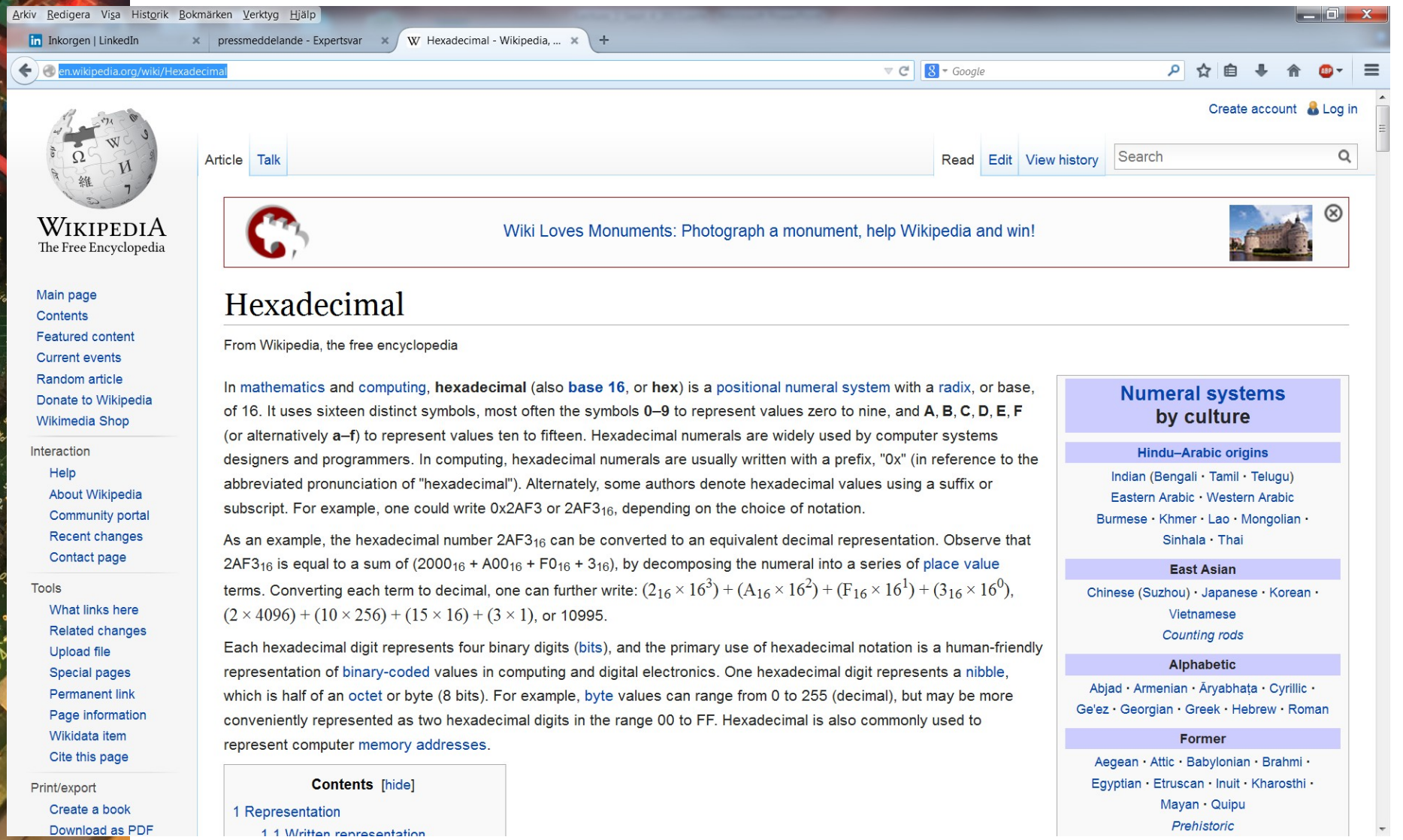
$1 * 16^5 + 14 * 16^4 + 13 * 16^3 + 0 * 16^2 + 2 * 16^1 + 1 * 16^0 =$

$1 * 1\,048\,576 + 14 * 65\,536 + 13 * 4096 + 0 * 256 + 2 * 16 + 1 * 1 =$

$2\,019\,361_{10}$

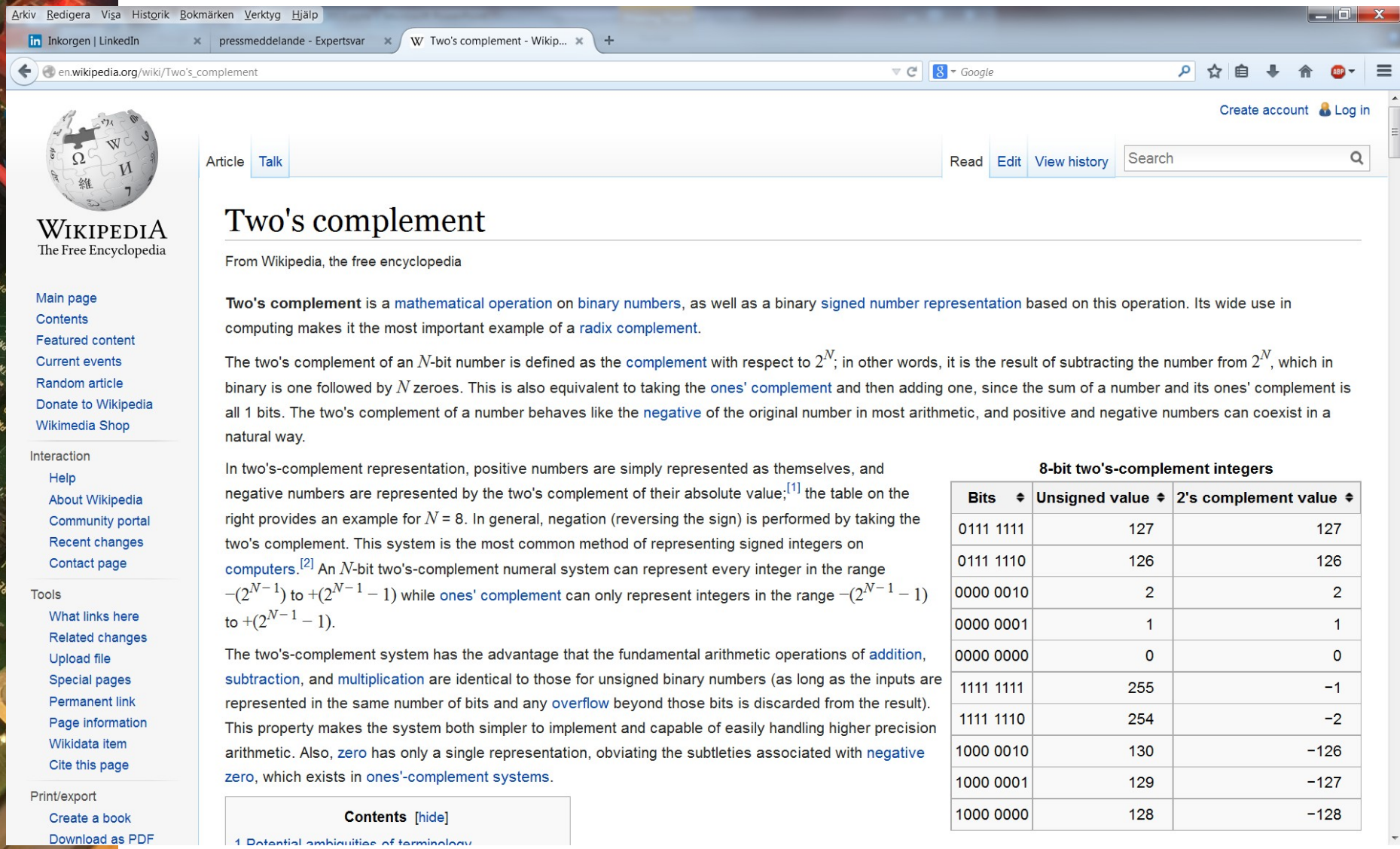
Hexadecimal numbers.

http://en.wikipedia.org/wiki/Hexadecimal



Two’s complement.

http://en.wikipedia.org/wiki/Two%27s_complement



Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp

Inkorgen | LinkedIn | pressmeddelande - Expertsvar | Two's complement - Wikip...

en.wikipedia.org/wiki/Two's_complement

Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF

Two's complement

From Wikipedia, the free encyclopedia

Two's complement is a mathematical operation on binary numbers, as well as a binary signed number representation based on this operation. Its wide use in computing makes it the most important example of a radix complement.

The two's complement of an N-bit number is defined as the complement with respect to 2^N; in other words, it is the result of subtracting the number from 2^N, which in binary is one followed by N zeroes. This is also equivalent to taking the ones' complement and then adding one, since the sum of a number and its ones' complement is all 1 bits. The two's complement of a number behaves like the negative of the original number in most arithmetic, and positive and negative numbers can coexist in a natural way.

In two's-complement representation, positive numbers are simply represented as themselves, and negative numbers are represented by the two's complement of their absolute value;[1] the table on the right provides an example for $N = 8$. In general, negation (reversing the sign) is performed by taking the two's complement. This system is the most common method of representing signed integers on computers.[2] An N-bit two's-complement numeral system can represent every integer in the range $-(2^{N-1})$ to $+(2^{N-1}-1)$ while ones' complement can only represent integers in the range $-(2^{N-1}-1)$ to $+(2^{N-1}-1)$.

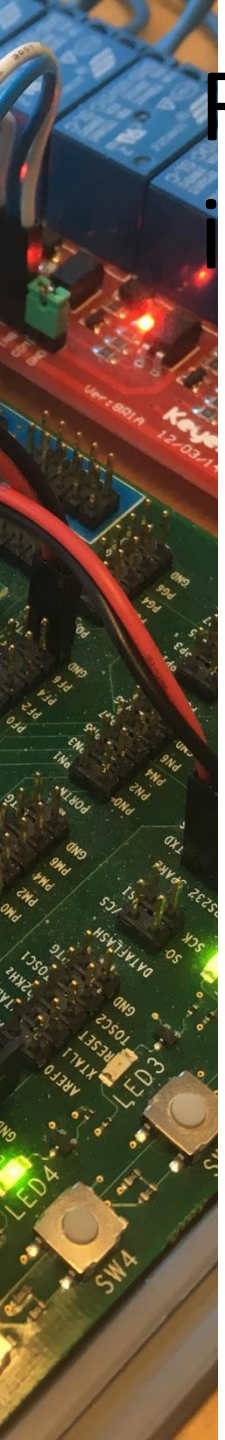
The two's-complement system has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers (as long as the inputs are represented in the same number of bits and any overflow beyond those bits is discarded from the result). This property makes the system both simpler to implement and capable of easily handling higher precision arithmetic. Also, zero has only a single representation, obviating the subtleties associated with negative zero, which exists in ones'-complement systems.

Contents [hide]

1 Potential ambiguities of terminology

8-bit two's-complement integers

Bits	Unsigned value	2's complement value
0111 1111	127	127
0111 1110	126	126
0000 0010	2	2
0000 0001	1	1
0000 0000	0	0
1111 1111	255	−1
1111 1110	254	−2
1000 0010	130	−126
1000 0001	129	−127
1000 0000	128	−128

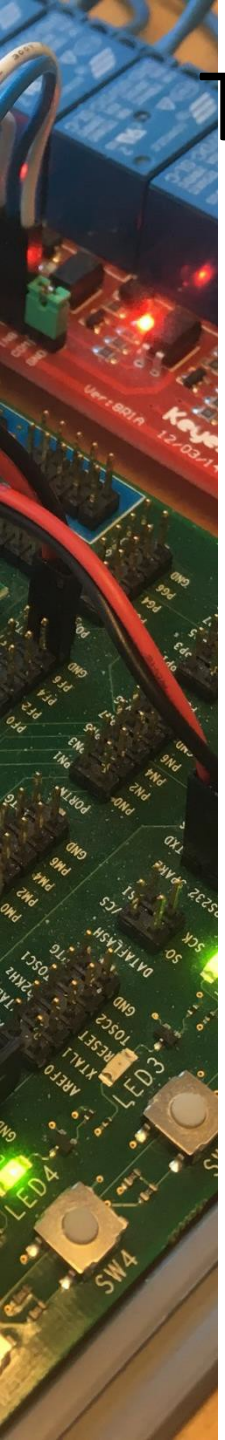


Representing negative numbers internally

- Using Most Significant Bit (MSB) as a sign bit
- $+36_{10}$ → $0010\ 0100_2$
  -36_{10} → $1010\ 0100_2$
- There are two ways to represent zero
  - 10000000_2 and 00000000_2
- Hard to use internally (arithmetics)

  →

  Encode negative numbers using Two's complement



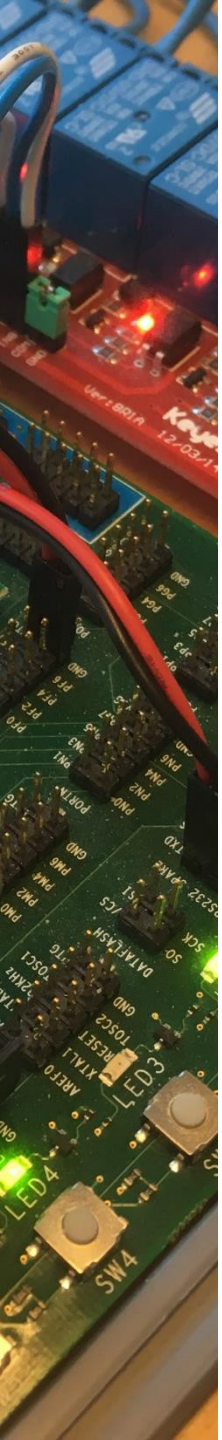
Two's complement

- Steps for encoding
  1. Remove sign bit
  2. Invert bits
  3. Add 1

Encoding example:
8-bit integer!

-36_{10} → $1010\ 0100_2$

1. 0010 0100
2. 1101 1011
3. 1101 1011+1=1101 1100 (the 9th bit is skipped if there is one)



Arithmetic example:

Add the two 8-bit integers -36_{10} and 10_{10}

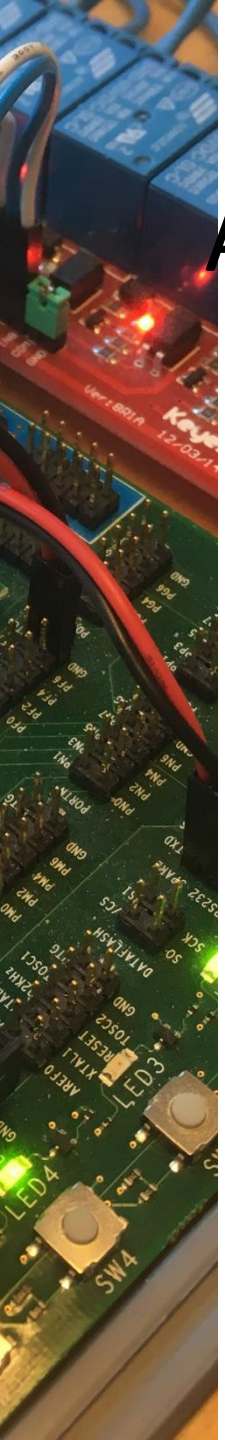
-36_{10} in two's complement → $1101\ 1100_2$

$+10_{10}$ → $0000\ 1010_2$

```
 1101 1100 (← -36)
+0000 1010 (← 10)
 1110 0110
```

Decode :

1. 0110 0110
2. 1001 1001
3. 1001 1001 + 1 = 1001 1010 = -26_{10}



doc2549_ATmega2560
ATMega2560, 8-bit Microcontroller

Features

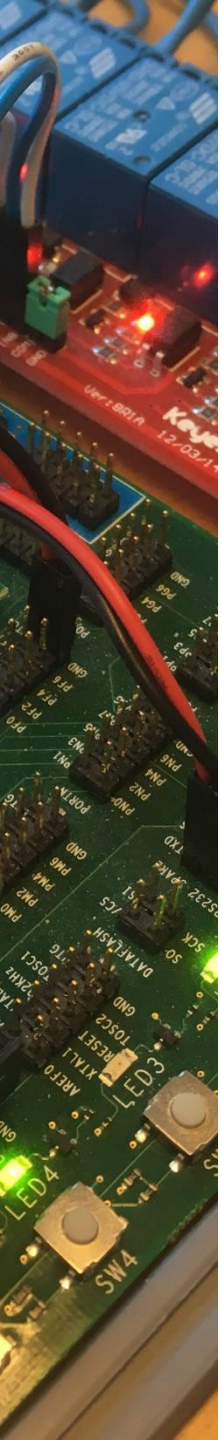
- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- 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

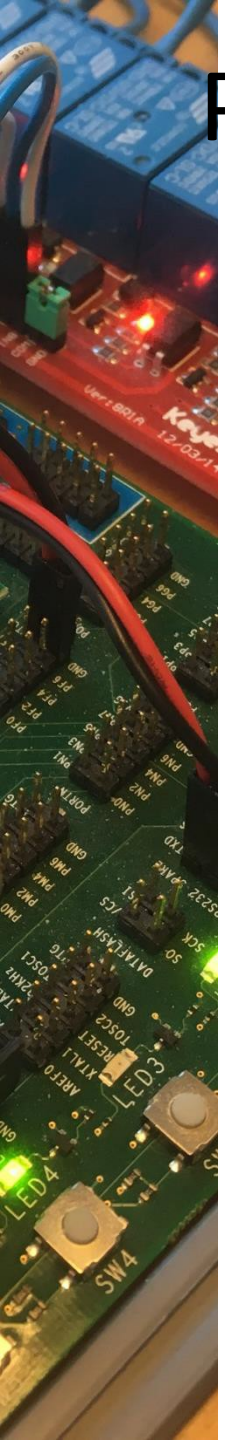
Registers and Operands

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

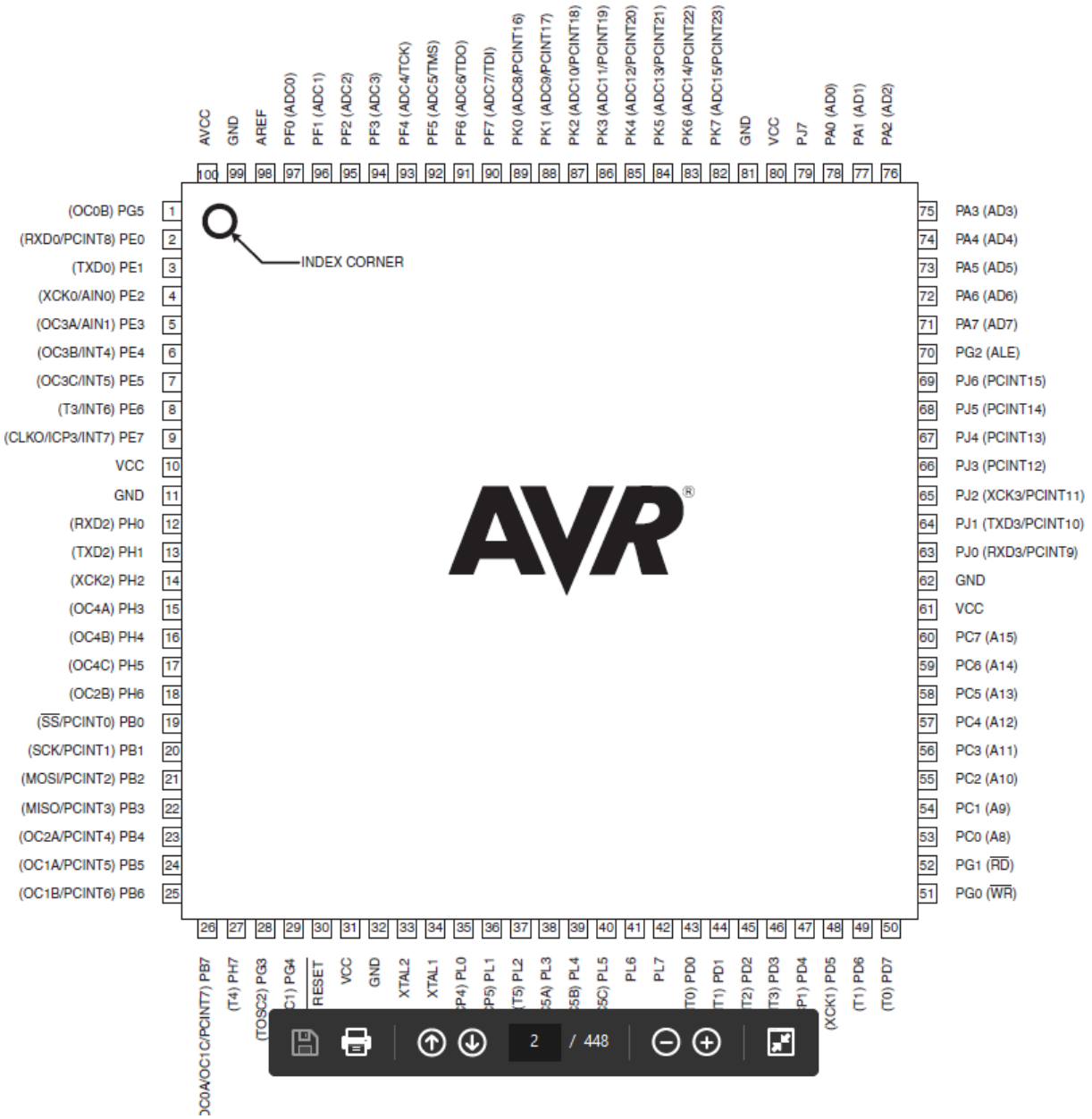
ATMEL®

8-bit AVR® Instruction Set

ATmega2560



Pin conf


AVCC 100
GND 99
AREF 98
PF0 (ADC0) 97
PF1 (ADC1) 96
PF2 (ADC2) 95
PF3 (ADC3) 94
PF4 (ADC4/TCK) 93
PF5 (ADC5/TMS) 92
PF6 (ADC6/TDO) 91
PF7 (ADC7/TDI) 90
PK0 (ADC8/PCINT16) 89
PK1 (ADC9/PCINT17) 88
PK2 (ADC10/PCINT18) 87
PK3 (ADC11/PCINT19) 86
PK4 (ADC12/PCINT20) 85
PK5 (ADC13/PCINT21) 84
PK6 (ADC14/PCINT22) 83
PK7 (ADC15/PCINT23) 82
GND 81
VCC 80
PJ7 79
PA0 (AD0) 78
PA1 (AD1) 77
PA2 (AD2) 76
(OC0B) PG5 1
(RXD0/PCINT8) PE0 2
(TXD0) PE1 3
(XCK0/AIN0) PE2 4
(OC3A/AIN1) PE3 5
(OC3B/INT4) PE4 6
(OC3C/INT5) PE5 7
(T3/INT6) PE6 8
(CLKO/ICP3/INT7) PE7 9
VCC 10
GND 11
(RXD2) PH0 12
(TXD2) PH1 13
(XCK2) PH2 14
(OC4A) PH3 15
(OC4B) PH4 16
(OC4C) PH5 17
(OC2B) PH6 18
(SS/PCINT0) PB0 19
(SCK/PCINT1) PB1 20
(MOSI/PCINT2) PB2 21
(MISO/PCINT3) PB3 22
(OC2A/PCINT4) PB4 23
(OC1A/PCINT5) PB5 24
(OC1B/PCINT6) PB6 25
INDEX CORNER
AVR
75 PA3 (AD3)
74 PA4 (AD4)
73 PA5 (AD5)
72 PA6 (AD6)
71 PA7 (AD7)
70 PG2 (ALE)
69 PJ6 (PCINT15)
68 PJ5 (PCINT14)
67 PJ4 (PCINT13)
66 PJ3 (PCINT12)
65 PJ2 (XCK3/PCINT11)
64 PJ1 (TXD3/PCINT10)
63 PJ0 (RXD3/PCINT9)
62 GND
61 VCC
60 PC7 (A15)
59 PC6 (A14)
58 PC5 (A13)
57 PC4 (A12)
56 PC3 (A11)
55 PC2 (A10)
54 PC1 (A9)
53 PC0 (A8)
52 PG1 (RD)
51 PG0 (WR)
26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
RESET VCC GND XTAL2 XTAL1
2 / 448

Architecture

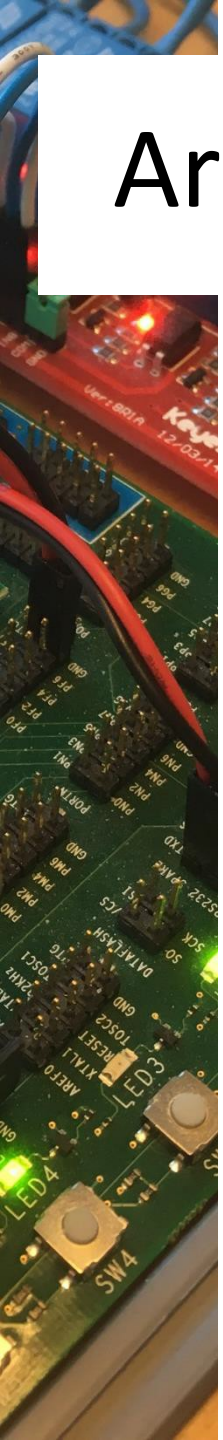
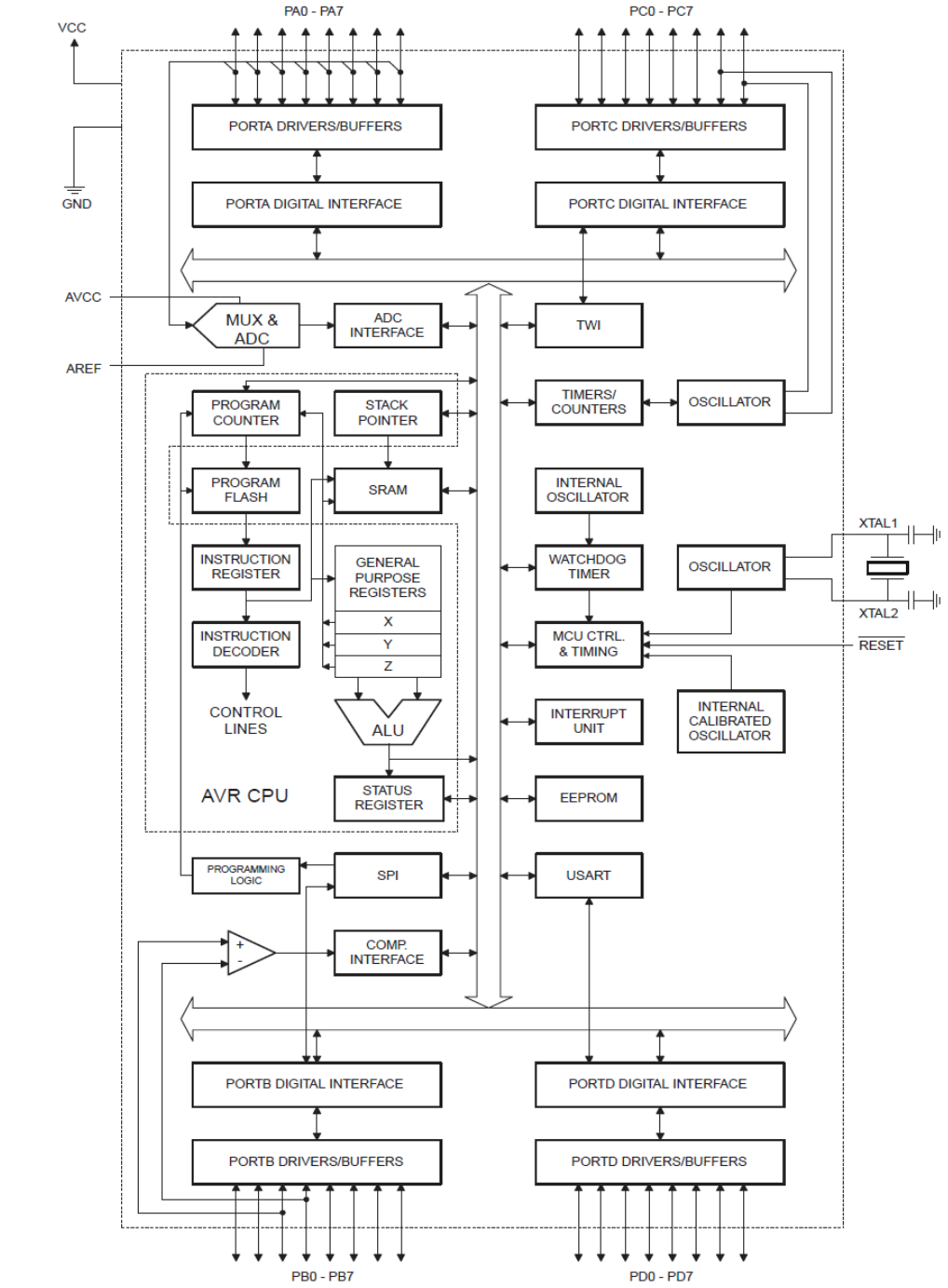


Figure 2. Block Diagram



AVR CPU Core

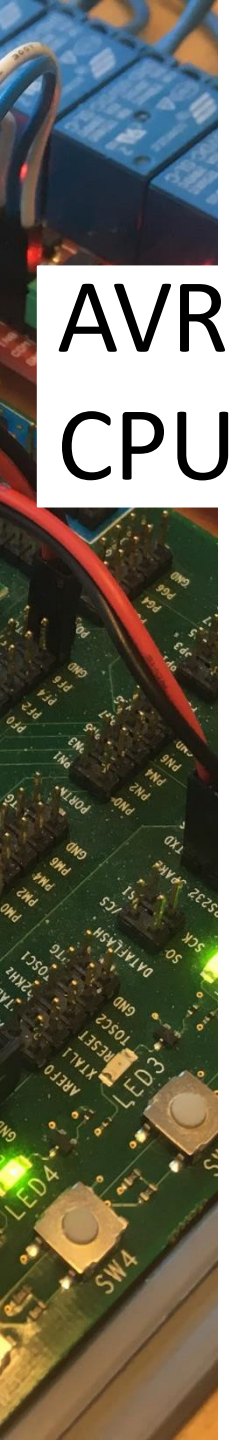
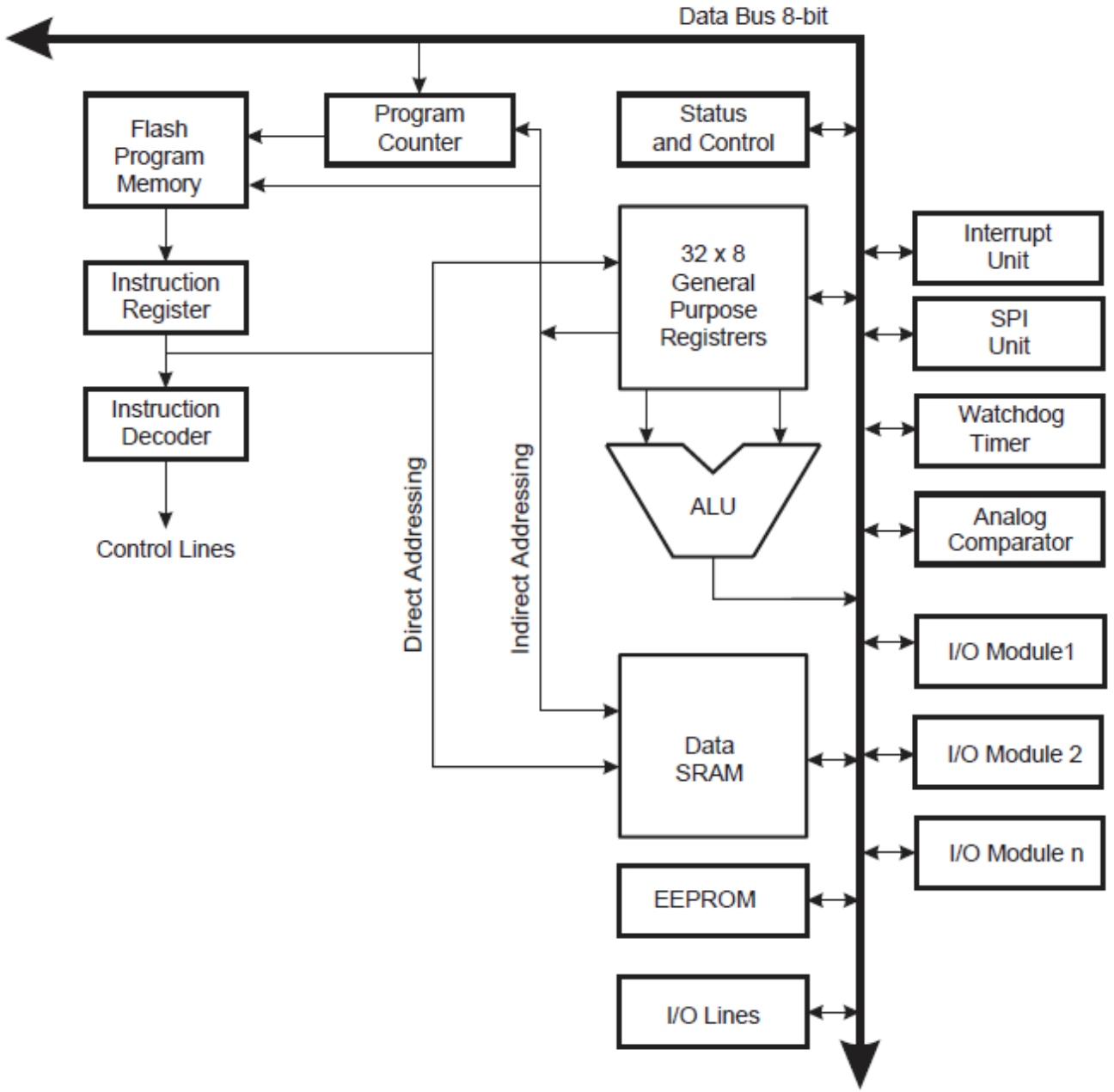
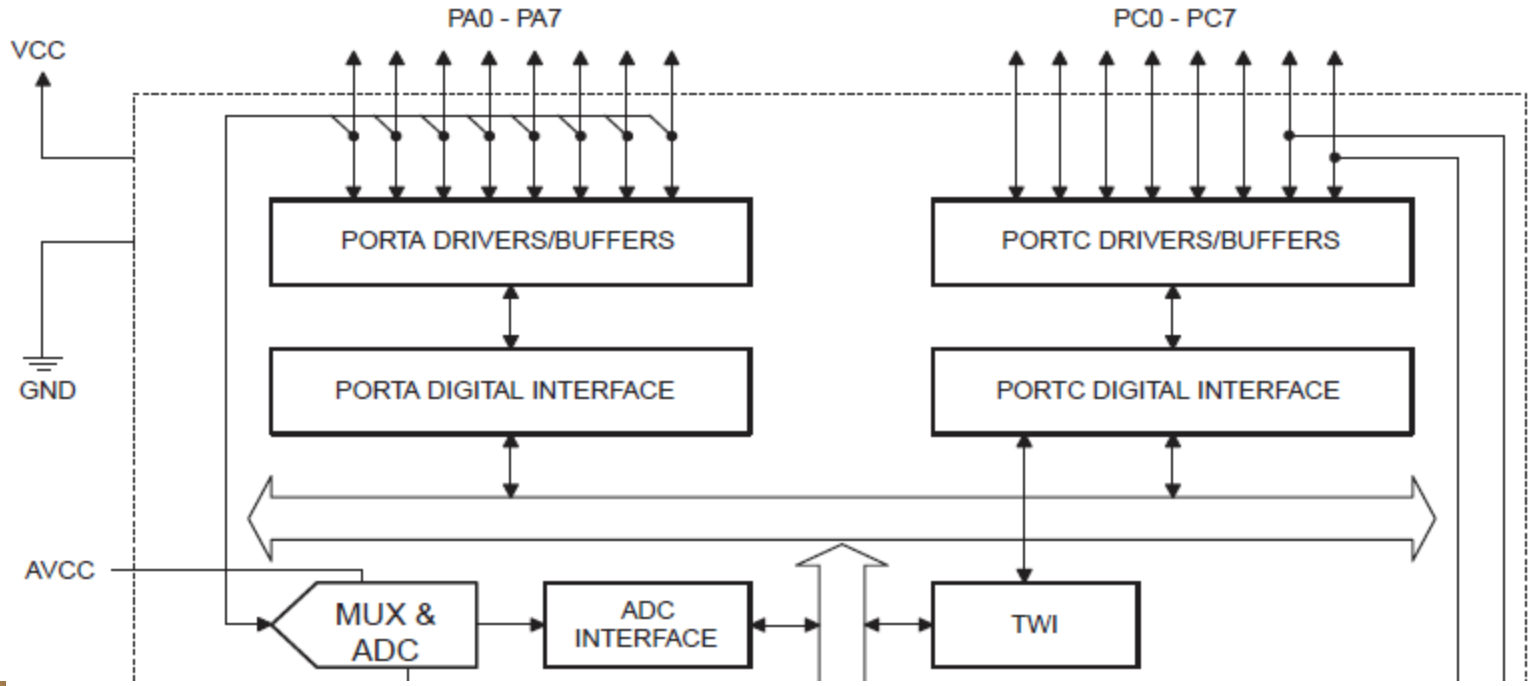


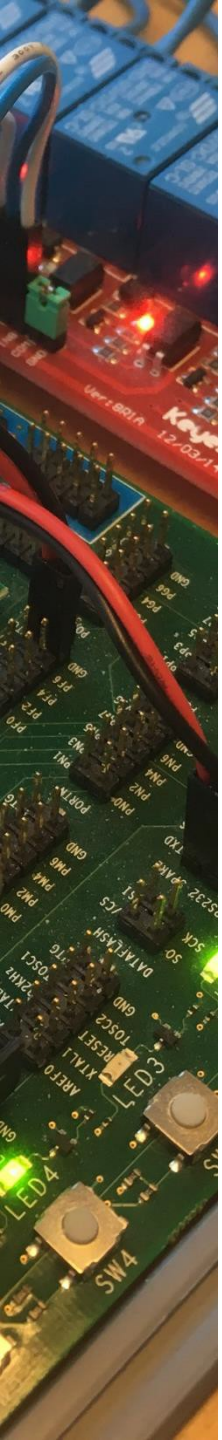
Figure 5. Block Diagram of the AVR MCU Architecture



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	

Data memory map 1

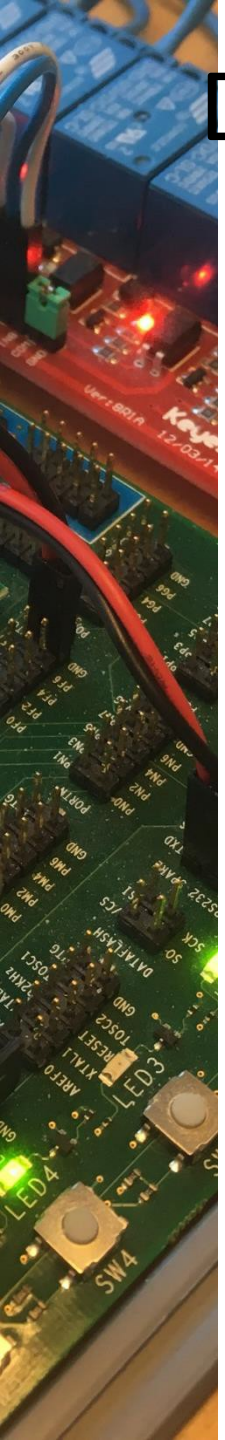
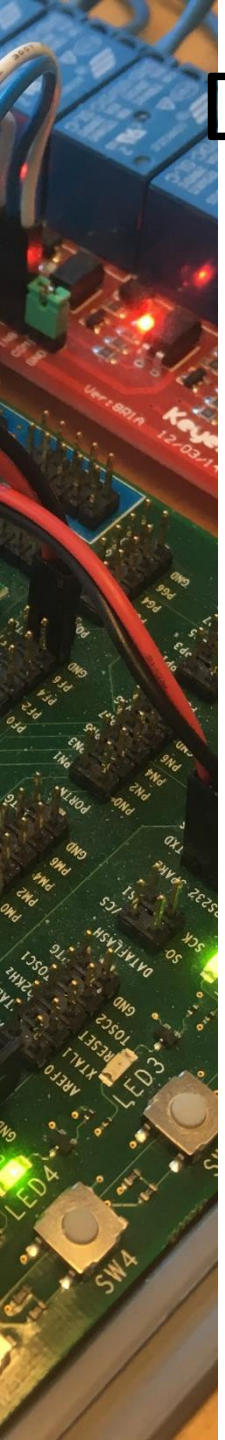
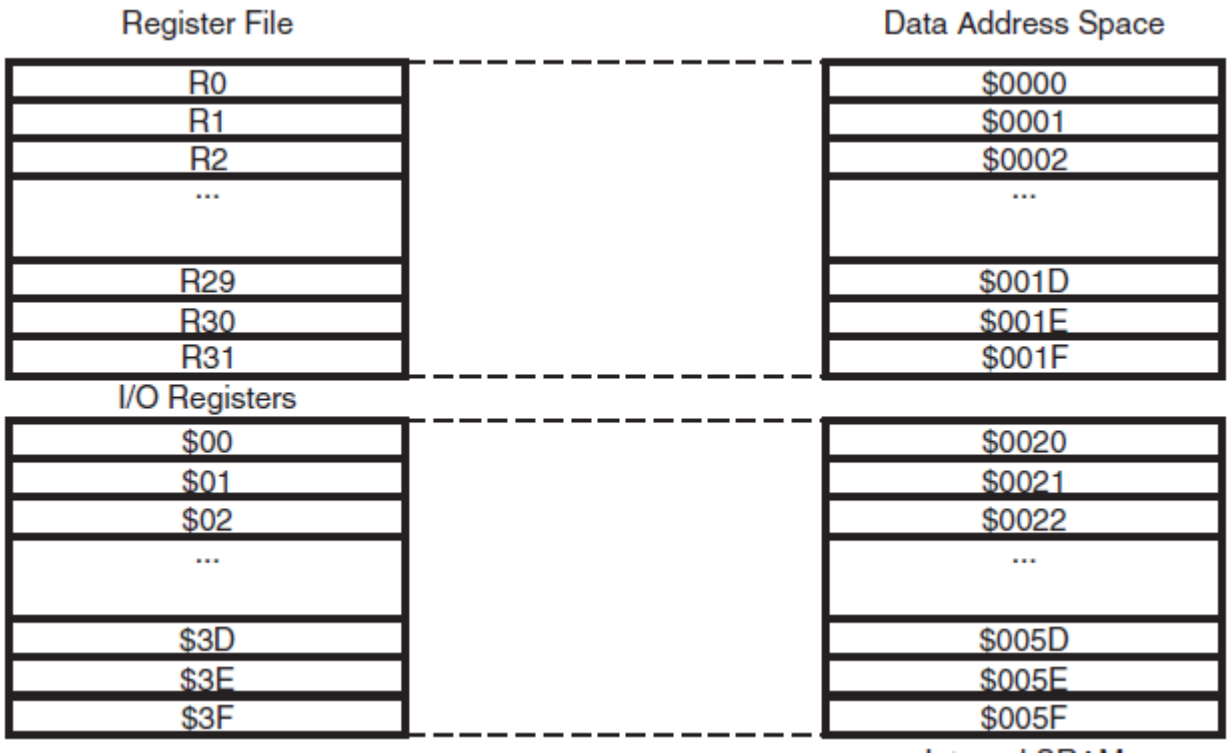


Figure 8-2. Data Memory Map

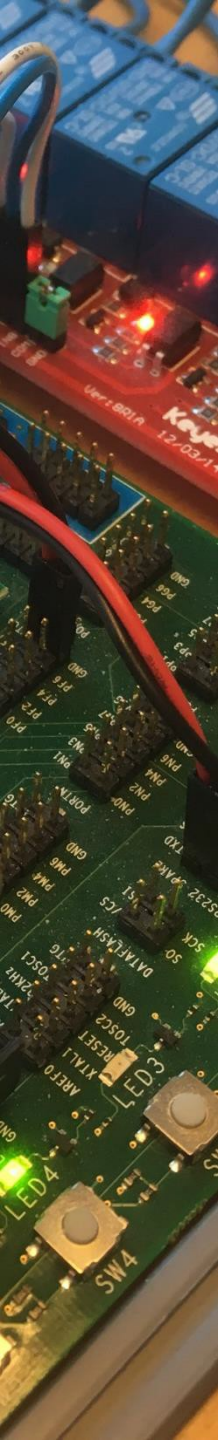
Address (HEX)	
0 - 1F	32 Registers
20 - 5F	64 I/O Registers
60 - 1FF	416 External I/O Registers
200 21FF	Internal SRAM (8192 × 8)
2200	External SRAM (0 - 64K × 8)
FFFF	

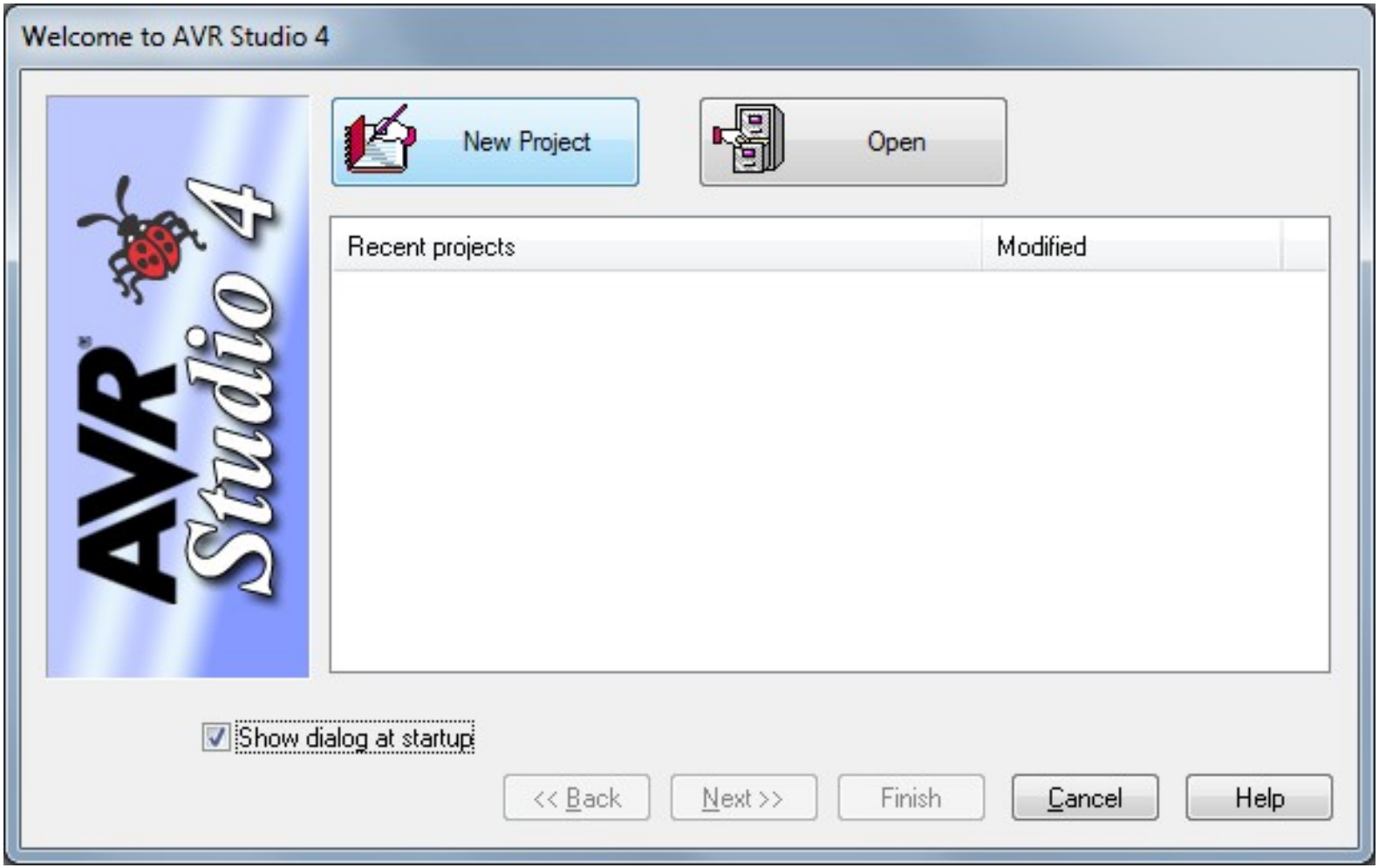
Data memory map 2



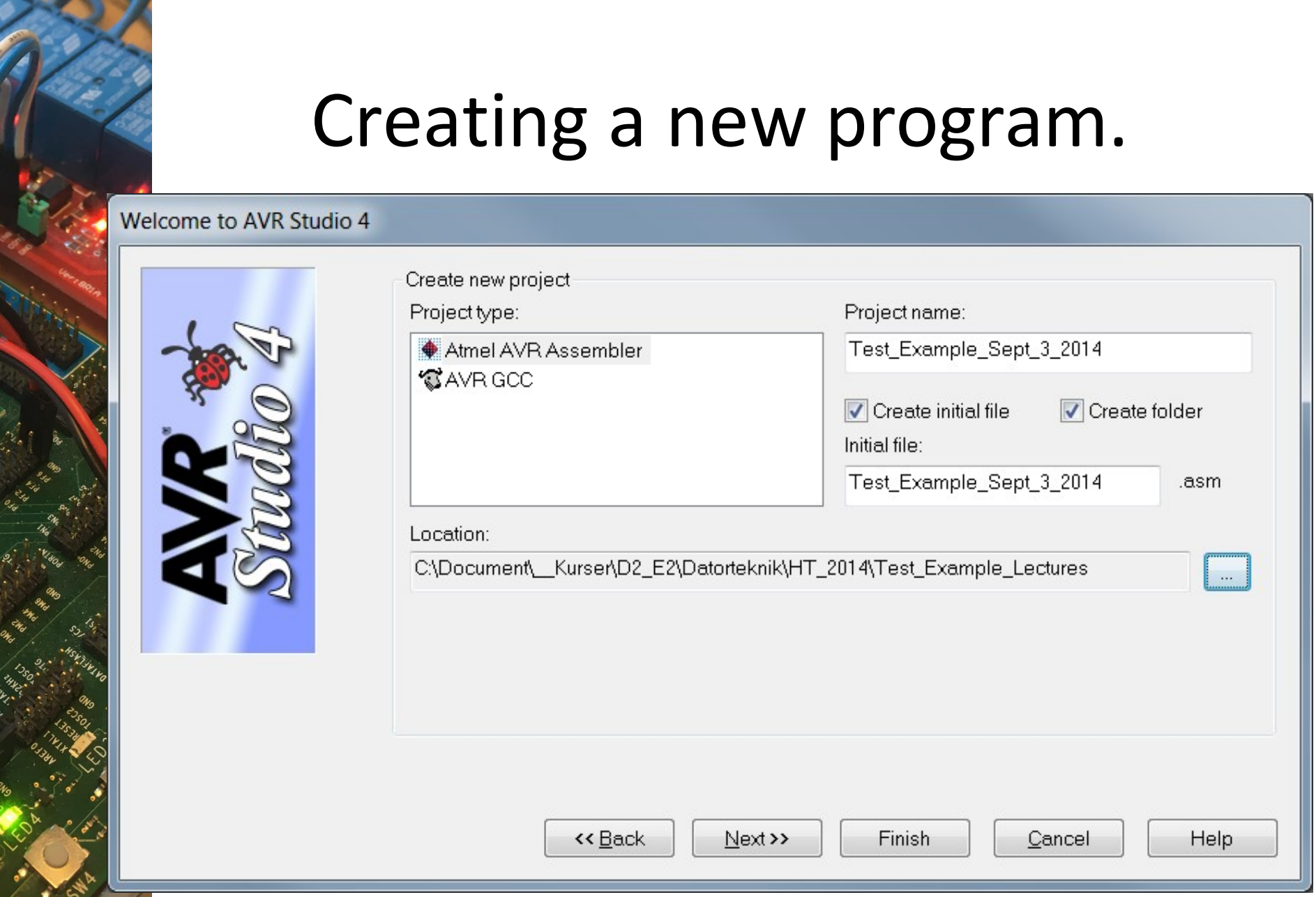


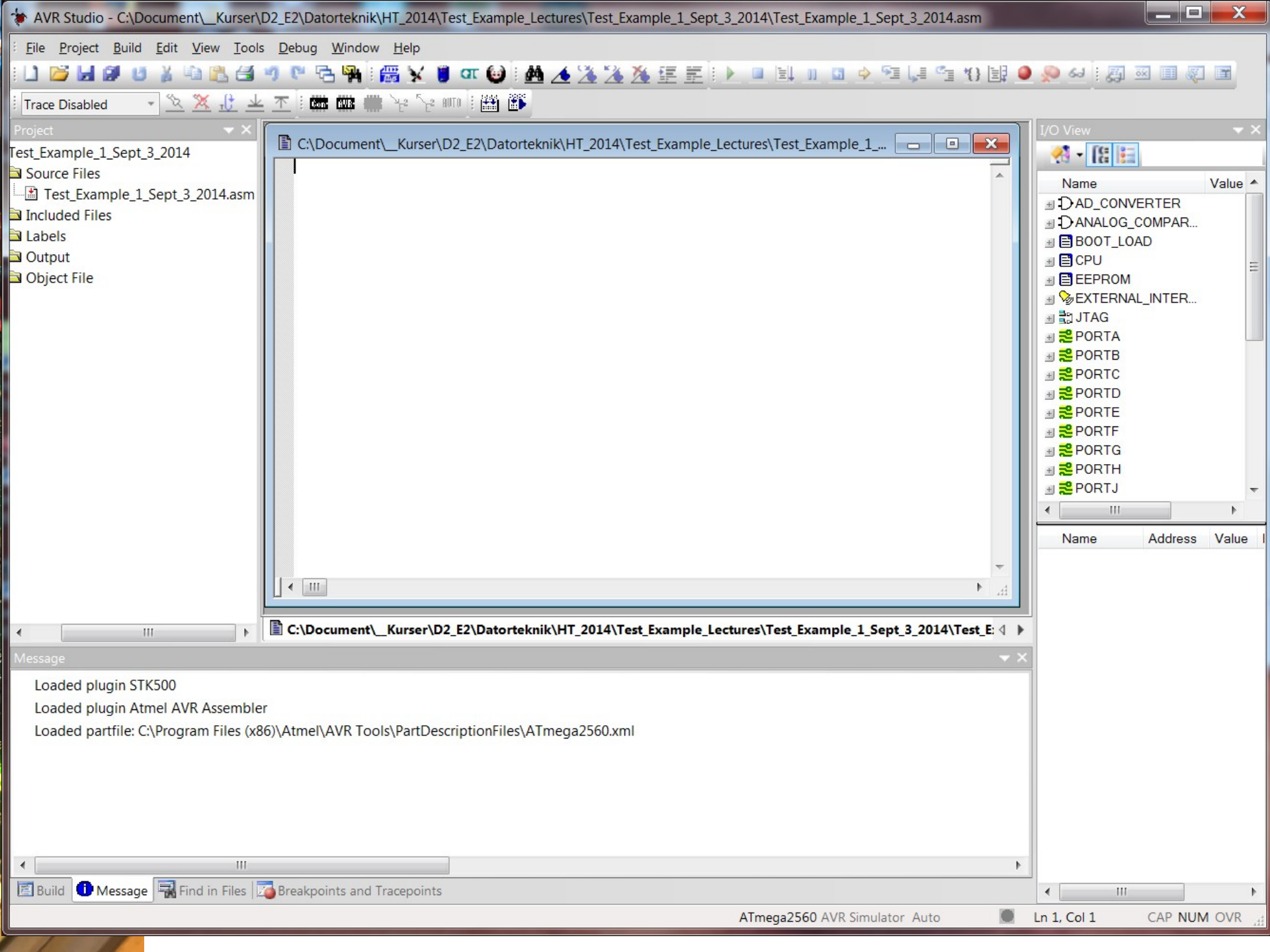
AVR Studio 4




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

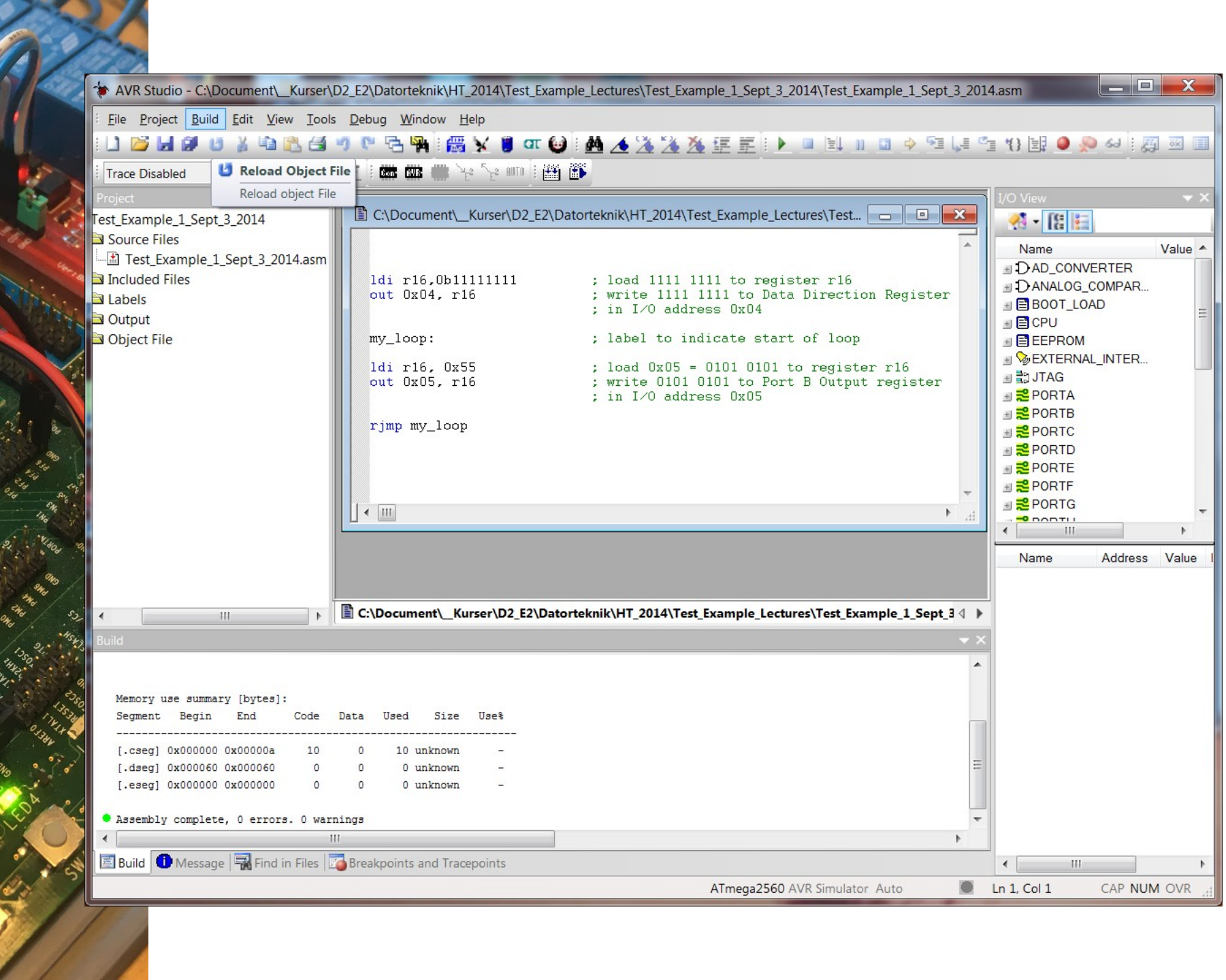
Creating a new program.

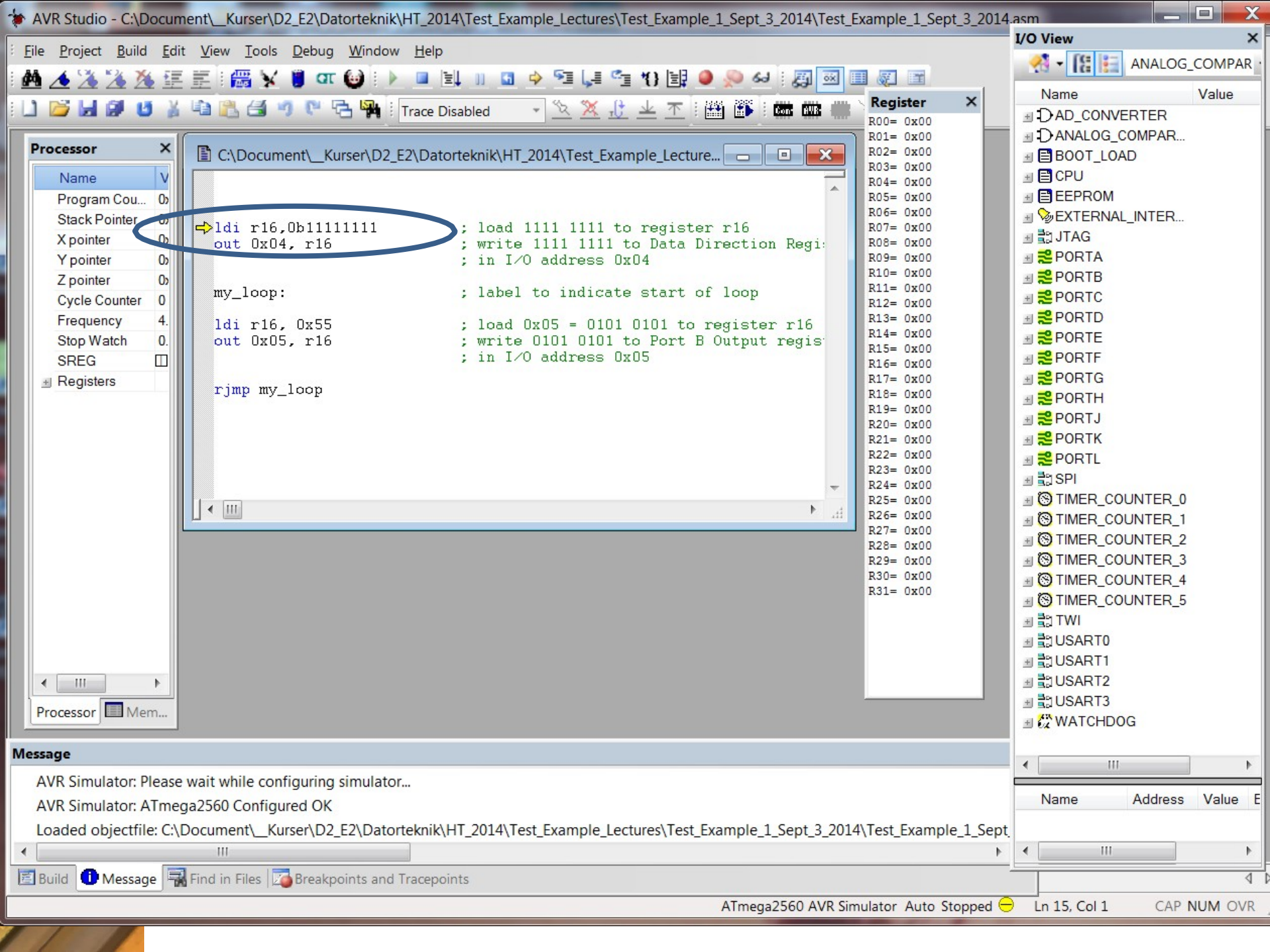


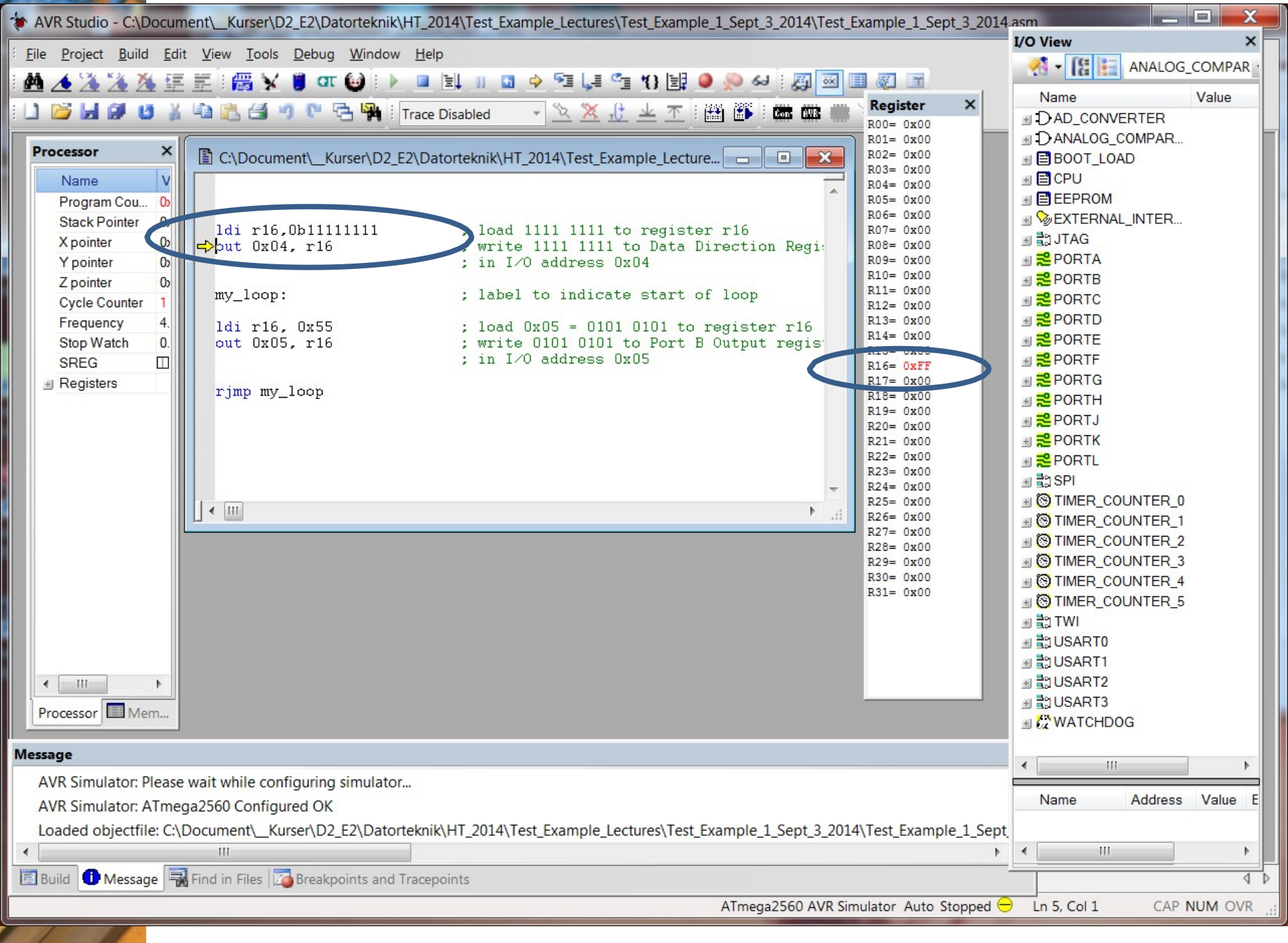

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
Test_Example_1_Sept_3_2014
Source Files
Test_Example_1_Sept_3_2014.asm
Included Files
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

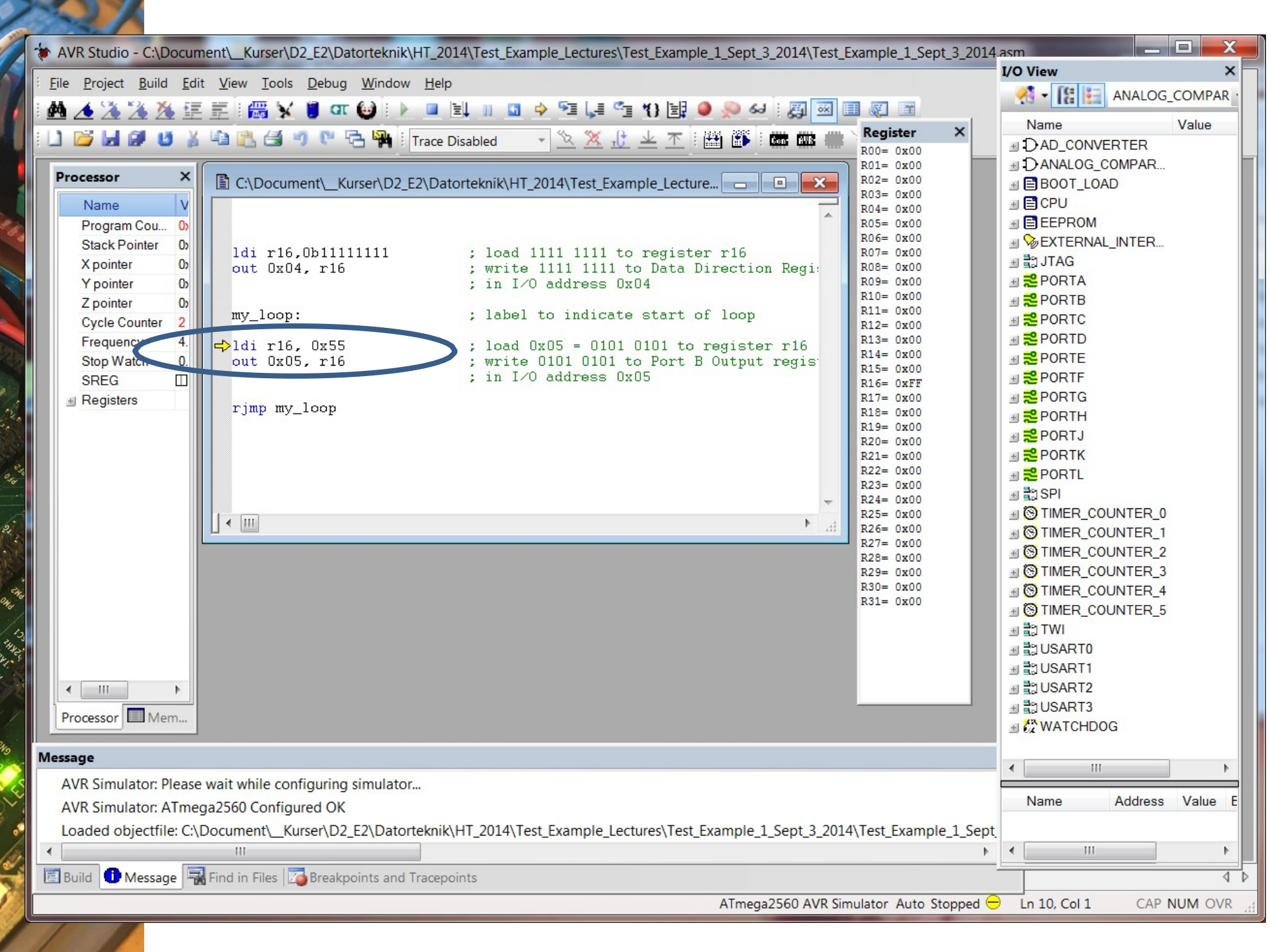
C:\Document_ Kurser\ D2_E2\ Datorteknik\HT_2014\ Test_Example_Lectures\ Test ..

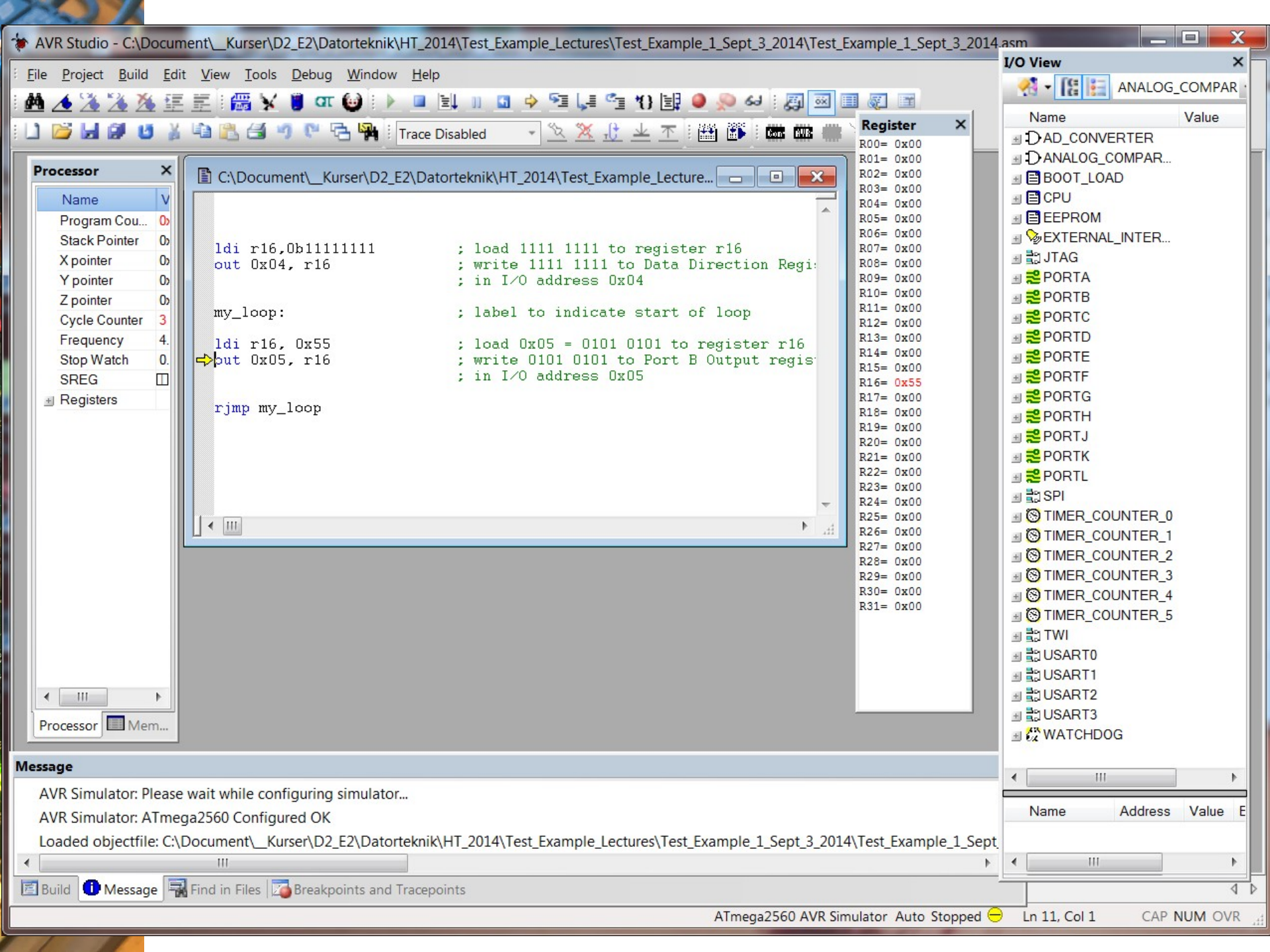
```
ldi r16,0bllllllll          ,   load 1111 1111 to register  r16
out  Ox04, r16              ,   write 1111 1111 to Data. Direction  Register
                            ,   1n I/0 address  Ox04

my_loop:                    ,   label to indicate start of  loop

ldi rl6, Ox55               ,   load Ox05 = 0101 0101 to register r16
out  Ox05, r16              ,   write 0101 0101 to Port B Output register
                            ,   1n I/0 address  Ox05

rJmp my_loop
```

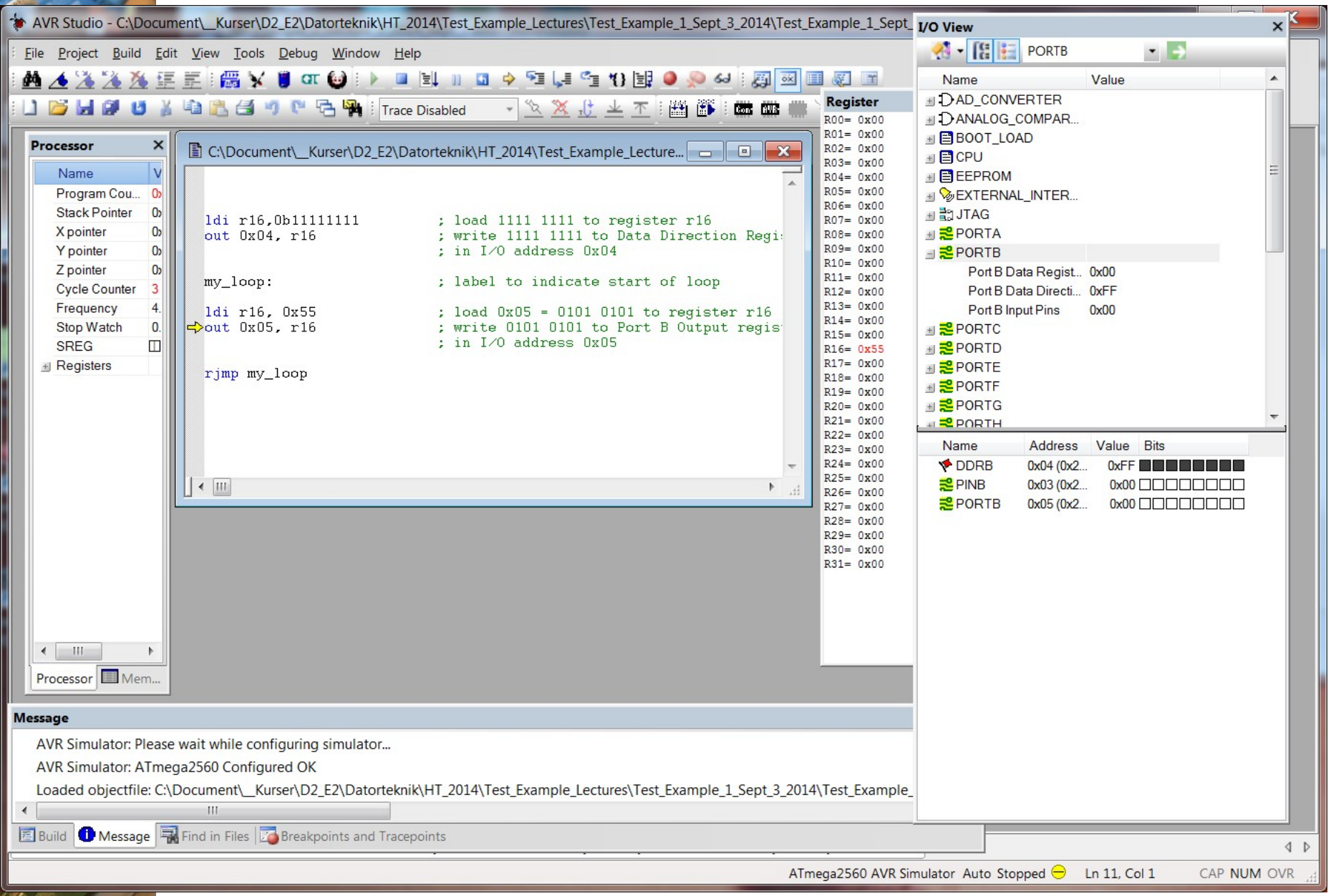

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Reload Object File
Reload object File
Project
Test_Example_1_Sept_3_2014
Source Files
Test_Example_1_Sept_3_2014.asm
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

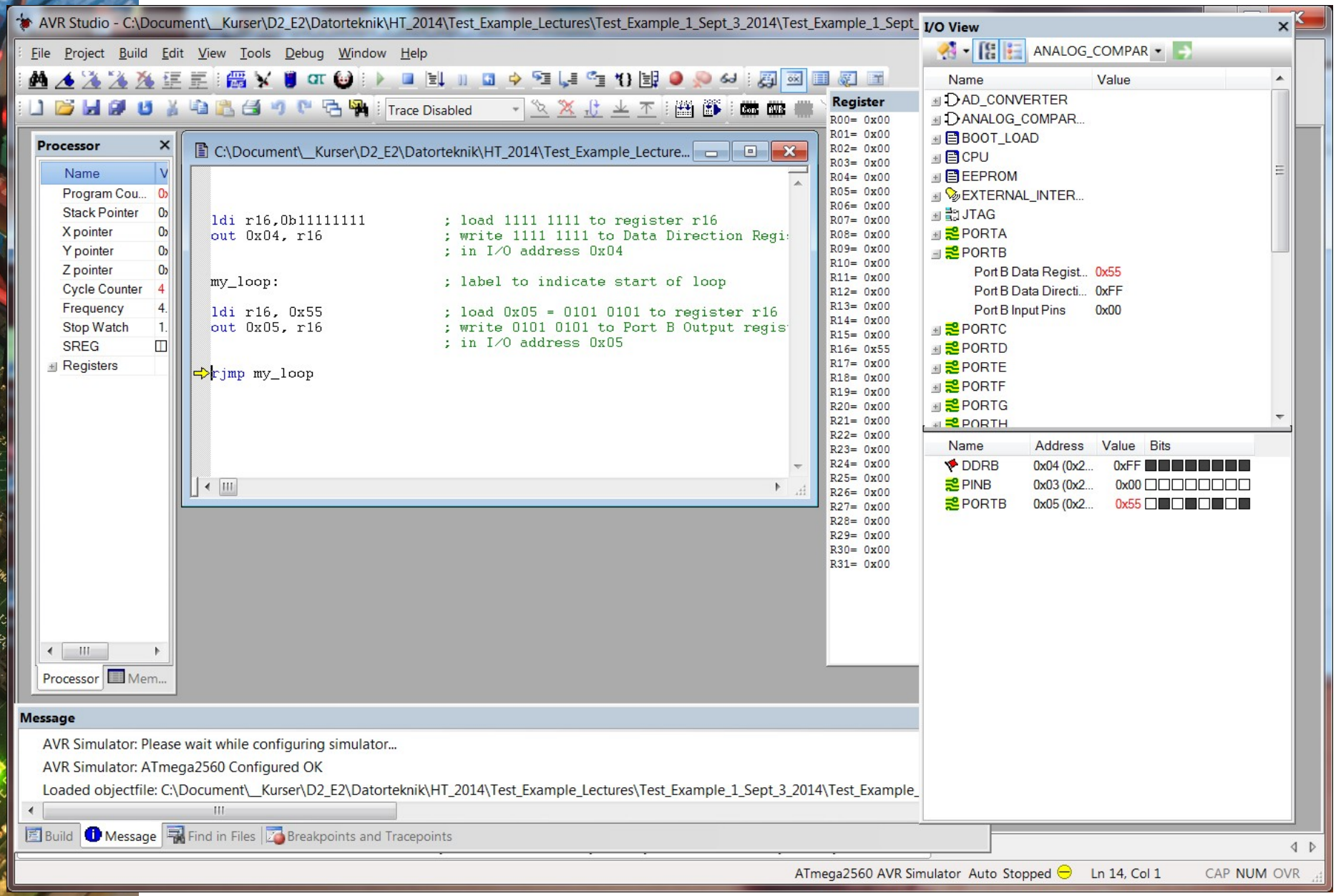

AVR Studio - C:\Document__Kurser\D2_E2\Datorteknik\HT_2014\Test_Example_Lectures\Test_Example_1_Sept_3_2014\Test_Example_1_Sept_3_2014.asm
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Processor
Name
Program Cou...
Stack Pointer
X pointer
Y pointer
Z pointer
Cycle Counter
Frequency
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
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]

10 Erase device before flash programming

10 Verify device after programming

Flash

(E) Use Current Simulator/Emulator FLASH Memory

@ Input HEX File C:\Documents and Settings\Kursers\D2_E2\Datorteknik\HT_2014\Laborationer D

[Program] [Verify] [Read]

EEPROM

(E) Use Current Simulator/Emulator EEPROM Memory

@ Input HEX File C:\Documents and Settings\Kursers\D2_E2\Datorteknik\HT_2013\Test_pro D

[Program] [Verify] [Read]

ELF Production File Format

Input ELF File: D

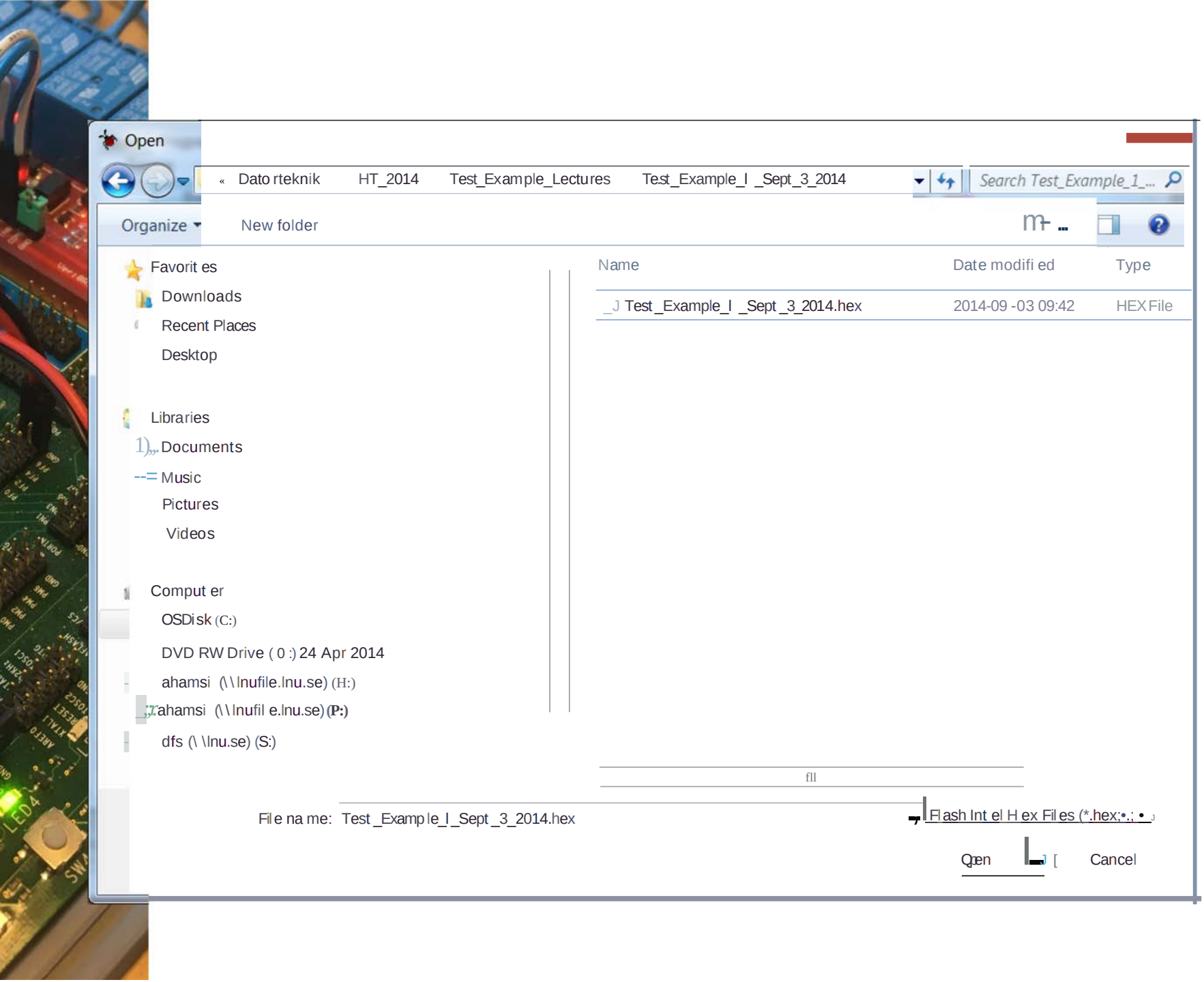
Save From: 10 FLASH 10 EEPROM ID FUSES ID LOCKBITS

Fuses and lockbits settings must be specified before saving to ELF

[Program] [Save]

Detecting on 'USB'...

STK600 with serial number 0D4A8D6873A0 found


Open
« Dato rteknik HT_2014 Test_Example_Lectures Test_Example_l _Sept_3_2014
Search Test_Example_1_...
Organize
New folder
Favorit es
Downloads
Recent Places
Desktop
Libraries
Documents
Music
Pictures
Videos
Comput er
OSDisk (C:)
DVD RW Drive (0 :) 24 Apr 2014
ahamsi (\\ lnufile.lnu.se) (H:)
ahamsi (\\ lnufil e.lnu.se) (P:)
dfs (\ \lnu.se) (S:)
Name
Date modifi ed
Type
Test _Example_l _Sept _3_2014.hex
2014-09 -03 09:42
HEX File
File na me: Test _Examp le_l _Sept _3_2014.hex
Flash Int el H ex Fil es (*.hex;•.; •
Open
Cancel

STK600 in ISP mode with ATmega2560

Main | Program | Fuses | Lock Bits | Advanced | HW Settings | HW Info | Auto |

Device

[Erase Device

0 Erase device before flash programming

0 Verify device after programming

Flash

Use Current Simulator/Emulator FLASH Memory

@Input HEX File _Example_I _Sept_3 _2014\Test_Example_I _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 D

[Program [Verify [Read

ELF Production File Format

Input ELF File: D

Save From: 0 FLASH 0 EEPROM D FUSES D LOCKBITS

[Program [Save

Fuses and lock bits settings must be specified before saving to ELF

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\...

```
;>>>> >>>>>>> >>>>>>>>  >>>>>>  >>>>>>>>> >>>>>
,     lDTr0l, 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 PORTE
,         Switches connected  to POPTA
;<<<<< <<<<<<<<<<< <<<<<<<<<<  <<<<<<<<<<


,  Set Data Direction Registers
 ldi rl6, 0xFF
 out 0x04, r16          ,  DDRE = 0x04
 out 0x0a, r16          ,  DDRd   0x0a

 ; output registers, PORTE and PORTD

 ldi rl6, 0x07
 out 0x05, r16          ,  PORTE   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)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Linn8universitetet

Task 1:

Write a program in Assembly language to light LED 2. You
can use any of the four ports, but start with PORTS.
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 , LEDS 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 LEDO then you press SW5.
For all other switches there should be no activity.

Task 4:

Run the program in Task 3 in the simula tor.

Task 5:

Task 5:

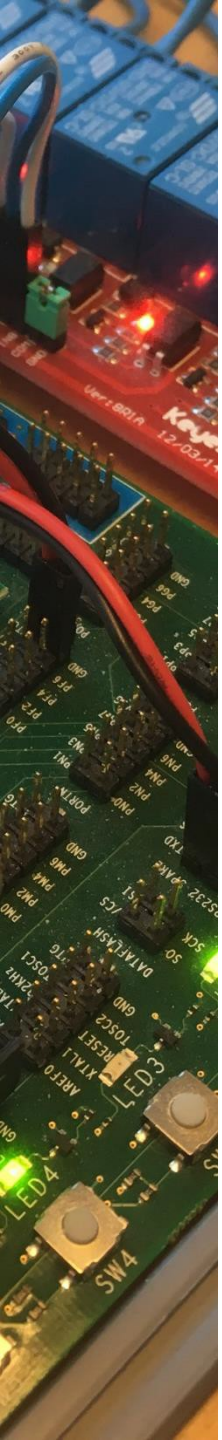
Write a program in Assemb ly language that creates a Ring Counter. The values should be displayed with the LEDs. Use shift instructions , LSL or LSR.
Make a delay of approximately 0.5 sec in between each count. Write the delay as a subroutine. For using the subroutine , you must initia lize the Stack 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
```

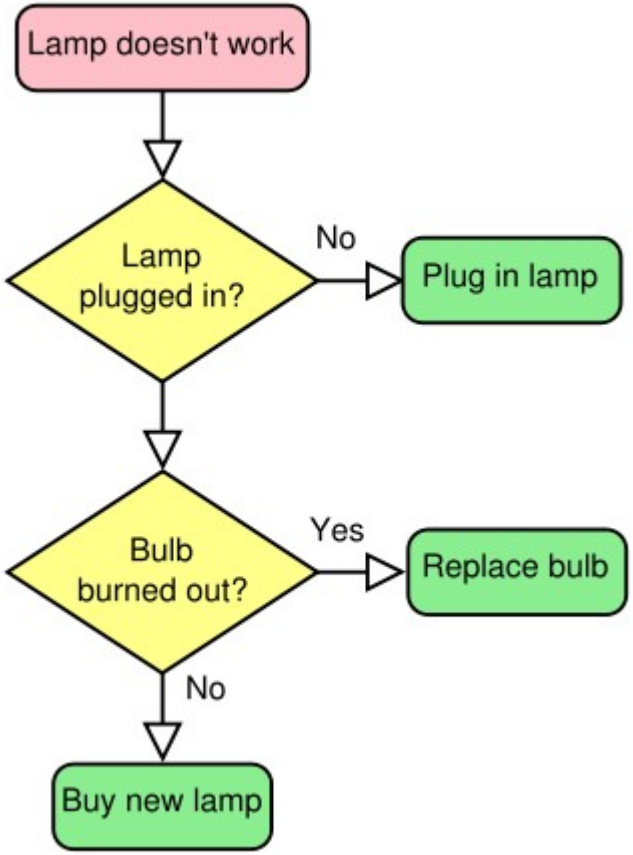
Funct ion, the 8 LEDs:
(0000 000X , 0000 00X0 , 0000 0X00, 0000 X000 , 000X 0000, 00X0 0000 , 0X00 0000, XOOO0000)

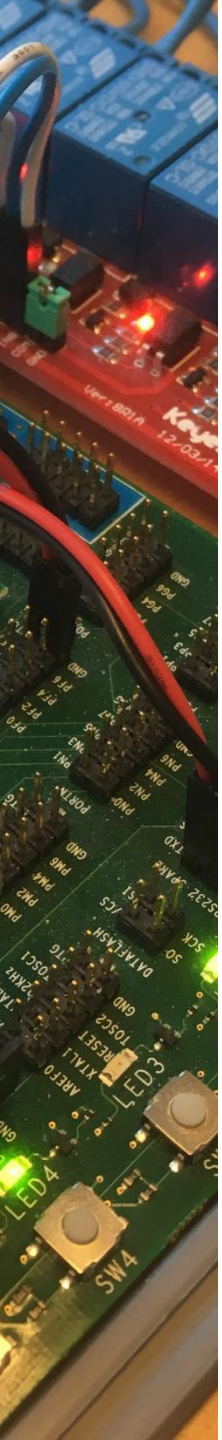
Task 6:
Write a program in Assembly language that creates a Johnson Counter in an infinite loop.

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)

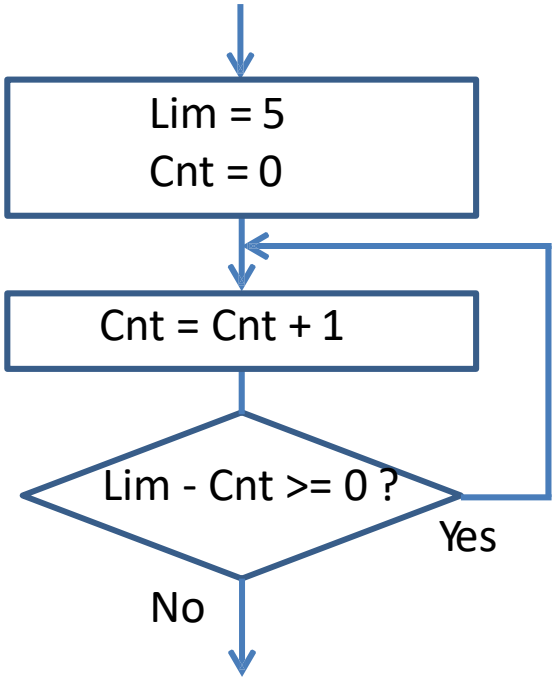


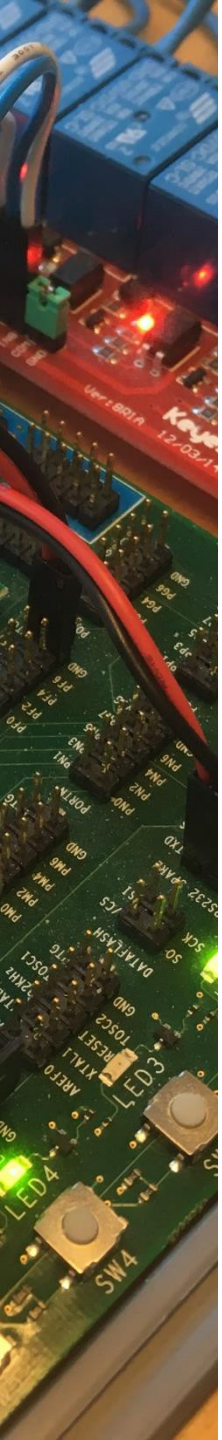
Using Flow Charts





Loop:

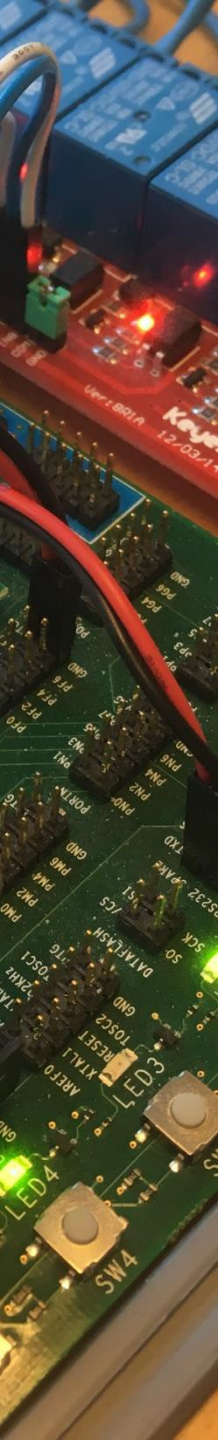




Simple delay loop

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
```



AVR Delay Loop Calculator

8.0 MHz microcontroller clock frequency

0 cycles for CALL/RET or other overhead

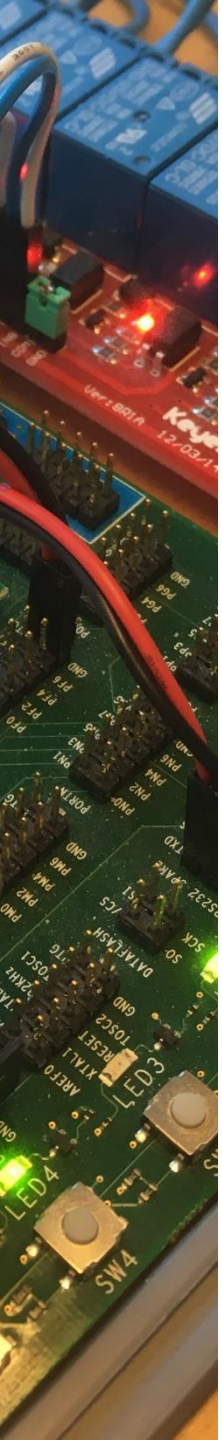
1 ns us ms s mins hrs days

8000 cycles go

(•) assembler ( ) avr-gcc

```
; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 8 000 cycles
; 1ms at 8.0 MHz

    ldi  r18, 11
    ldi  r19, 99
L1: dec  r19
    brne L1
    dec  r18
    brne L1
```

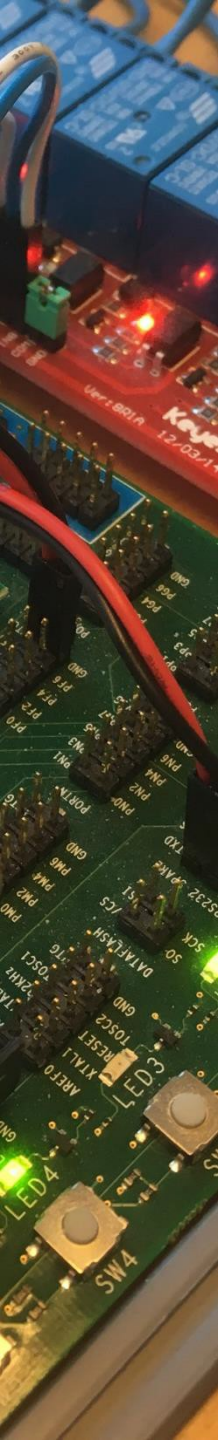
http://bretmulvey.com/avrdelay.html



Program example

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

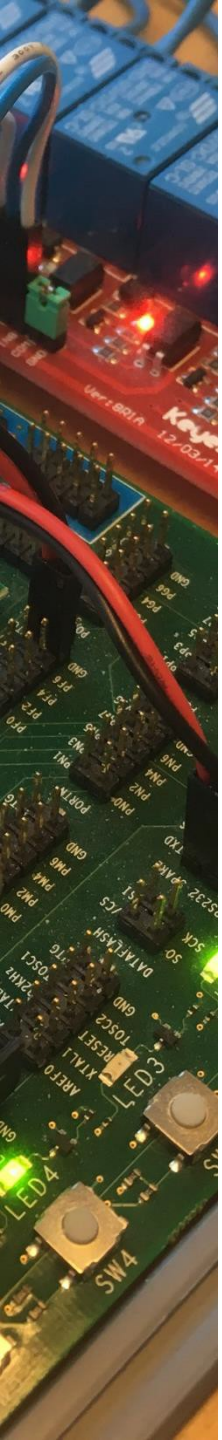
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;   1DT301, Computer Technology I
;   Date: 2015-09-03
;   Author:
;       Anders Haggren
;   Function:
;       Set output
;
;       Example #2
;       Lecture example 2015-09-03, lecture #2
;       Relay card connected to PORTD
;       LEDs connected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

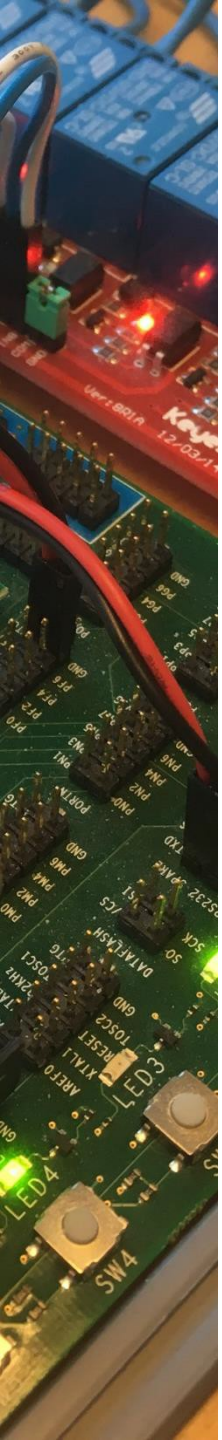
V: $Rd7 \bullet \overline{Rr7} \bullet \overline{R7} + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

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.

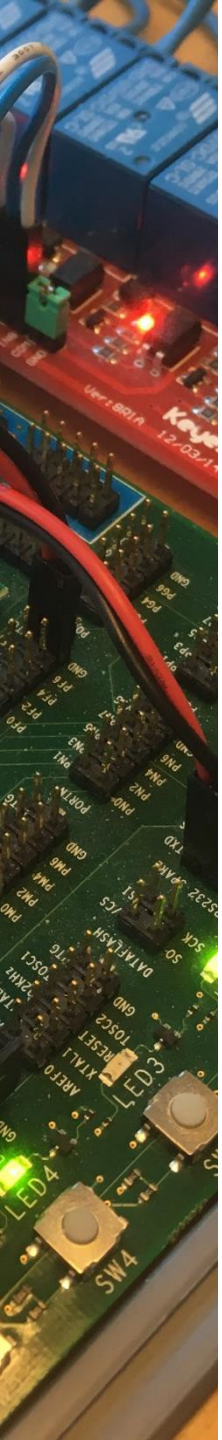
V: $Rd7 \bullet \overline{K7} \bullet \overline{R7} + \overline{Rd7} \bullet K7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

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.



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

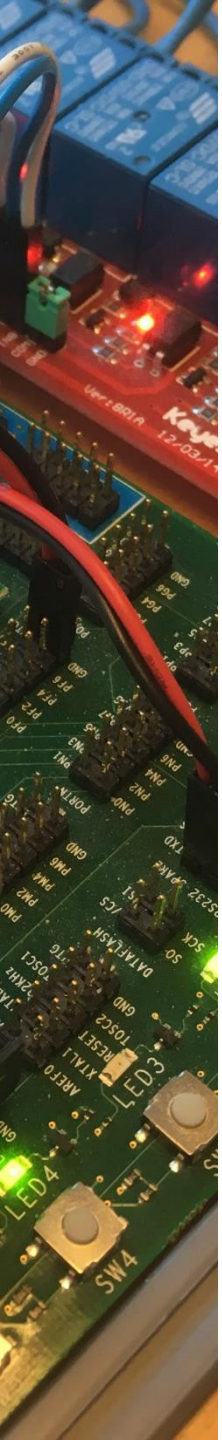
Status Register (SREG) and Boolean Formula:

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

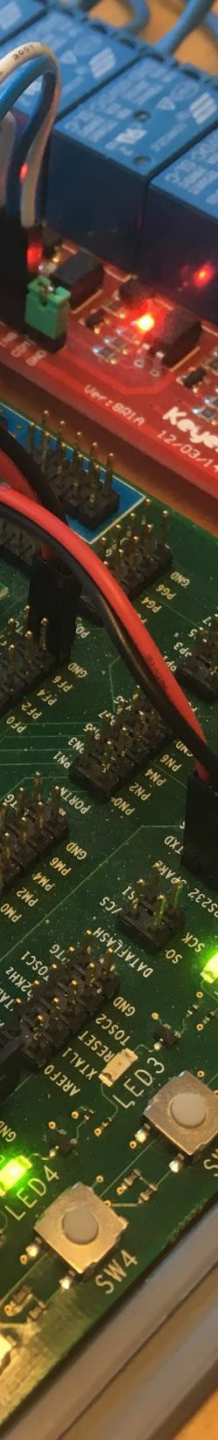
Status Register (SREG) and Boolean Formula:

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