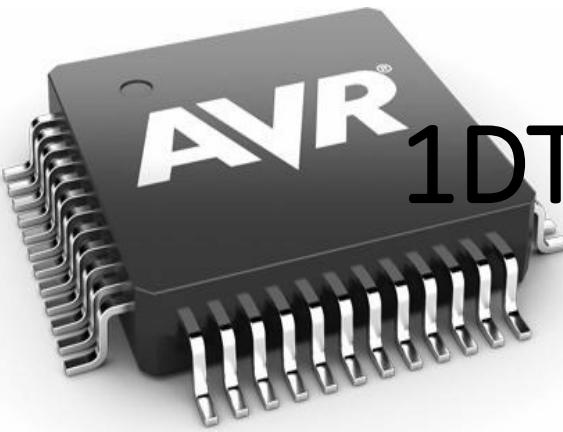


Welcome to
1DT301, Computer Technology
autumn 2019!

Schema

v36	Måndag 2019-09-02	Tisdag 2019-09-03	Onsdag 2019-09-04	Torsdag 2019-09-05	Fredag 2019-09-06
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

v 37	Måndag 2019-09-09	Tisdag 2019-09-10	Onsdag 2019-09-11	Torsdag 2019-09-12	Fredag 2019-09-13
8	08:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821		08:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1136A_V, Ma4052K Mötesrum E-möte, Anders Haggren, Föreläsning, 6821	08:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821	
9		10:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1136A_V, Ma4052K Mötesrum E-möte, Anders Haggren, Föreläsning, 6821			10:00 1DT301, TGI1V, HT19-P4021, HT18-61023, Ma4092K Grupprum, Laboration, 6820
10	Lab group 1, Monday morning.	10:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1136A_V, Ma4052K Mötesrum E-möte, Anders Haggren, Föreläsning, 6821		Lab group 5, Thursday morning.	Lab group 7 in Kalmar, Friday 10 – 14.
11					
12					
13	12:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821	12:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821	12:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821	12:15 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821	14:00
14	Lab group 2, Monday afternoon.	Lab group 3, Tuesday afternoon.	Lab group 4, Wednesday afternoon.	Lab group 6, Thursday afternoon.	
15					
16					
17					2019-09-12 13:15 - 17:00 1DT301, NGDNS, NGDPV, TGI1D-E, TGI1V, HT19-02283, HT19-F0273, HT19-P3283, HT19-P4021, HT19-U6026, HT18-61002, HT18-61003, HT18-F1600-1601, HT18-61004, HT18-61005, HT18-61021, HT18-61022, HT18-61023, D1166V, Amanuens 4, Laboration, 6821 ID 761560



1DT301, Computer Technology

- Programming in Assembly and C
- STK500 or STK600
- Lectures
- Laboratory work (6 labs)
- Laboratory work is mandatory
- Written exam,
Monday, October 28, 2019, 8.00-13.00
(2nd exam: Saturday, December 7.)



Planning, 1DT301, Computer Technology, autumn 2019. /Anders Haggren.

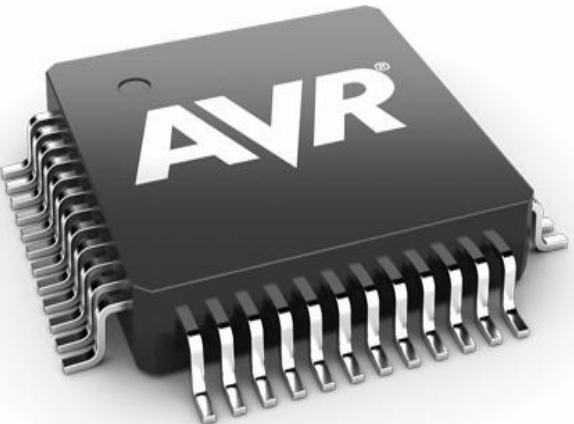
Lec.	Day:	Date:	Time:	Room:	Content:	Lab:
F1	Tuesday	Sept. 3	10 ¹⁵ – 12 ⁰⁰	D1136	Introduction, Course start Repetition of binary and hexadecimal numbers, two's complement form. Experimental environment, STK600, How to use ports. Assembly programming. Introduction to lab1.	
F2	Wednesday	Sept. 4	08 ¹⁵ – 12 ⁰⁰	D1136		
F3	Tuesday	Sept. 10	10 ¹⁵ – 12 ⁰⁰	D1136	Subroutine call, function of the STACK.	Lab 1. How to use the PORTs. Digital input/output.
F4	Wednesday	Sept. 11	08 ¹⁵ – 12 ⁰⁰	D1136	Using register pairs, subroutine call, random generator, Introduction to lab2.	
F5	Tuesday	Sept. 17	10 ¹⁵ – 12 ⁰⁰	D1136	Atmel AVR Assembler. Jumping and branching	Lab 2. Subroutines
F6	Wednesday	Sept. 18	08 ¹⁵ – 12 ⁰⁰	D1136	Using interrupts. Introduction to lab3.	
F7	Tuesday	Sept. 24	10 ¹⁵ – 12 ⁰⁰	D1136	Using timers and serial communication, USART, Introduction to lab4.	Lab 3. Interrupt routines
C1	Wednesday	Sept. 25	08 ¹⁵ – 12 ⁰⁰	D1136	<i>Introduction to C-programming language. Lars Karlsson.</i>	
F8	Tuesday	Oct. 1	10 ¹⁵ – 12 ⁰⁰	D1136	Interfacing to Hitachi display. Introduction to lab5.	Lab 4. Timer and UART.
C2	Wednesday	Oct. 2	08 ¹⁵ – 12 ⁰⁰	D1136	<i>Introduction to C-programming language. Lars Karlsson.</i>	
F9	Tuesday	Oct. 8	10 ¹⁵ – 12 ⁰⁰	D1136	Disassembler, instructions op-code.	Lab 5. LCD-Display.
C3	Wednesday	Oct. 9	08 ¹⁵ – 12 ⁰⁰	D1136	<i>Introduction to C-programming language. Lars Karlsson.</i> Introduction to lab6.	
C4	Tuesday	Oct. 15	10 ¹⁵ – 12 ⁰⁰	D1136	<i>Introduction to C-programming language. Lars Karlsson.</i>	Lab 6. CyberTech Wall
F10	Wednesday	Oct. 16	08 ¹⁵ – 12 ⁰⁰	D1136	Addressing modes, program examples.	Display and C-programming.
F11	Tuesday	Oct. 22	10 ¹⁵ – 12 ⁰⁰	D1136	Calculations with multiply, program examples.	
F12	Wednesday	Oct. 23	08 ¹⁵ – 12 ⁰⁰	D1136	Repetition and program examples.	
	Monday	Oct. 28	08 ⁰⁰ – 13 ⁰⁰		Written exam.	

Manuals and free on-line books:

- 1) [doc2466_ATmega16.pdf](#) - 8-bit Atmel Microcontroller with 4K/128K/256K Bytes In-System Programmable Flash
- 2) [Instruction set.pdf](#) – ATMEL 8-bit AVR Instruction Set
- 3) [doc1022_AVR_Assembler_user_guide.pdf](#) - AVR Assembler User Guide
- 4) [JHD202C_display_2x20.pdf](#)
- 5) **Beginners Introduction to the Assembly Language of ATMEL-AVR-Microprocessors** (English, Free on-line book),
www.avr-asm-download.de/beginner_en.pdf

Text books:

- 1) **Programming and Interfacing ATMEL's AVR's, 1st Edition, chapter 2, page 29 – 166** (English)
- 2) **Mikroprocessorteknik, Per Foyer, Studentlitteratur. ISBN 91-44-03876-3** (Swedish)



Assembly or Assembler language.

Assembly language

From Wikipedia, the free encyclopedia

An **assembly language** (or **assembler language**),^[1] often abbreviated **asm**, is any **low-level programming language** in which there is a very strong correspondence between the instructions in the **language** and the **architecture's machine code instructions**.^[2] Assembly language may also be called *symbolic machine code*.^{[3][4]}

Assembly code is converted into executable machine code by a **utility program** referred to as an **assembler**. The conversion process is referred to as **assembly**, as in **assembling** the **source code**. Assembly language usually has one statement per machine instruction (1:1), but **comments** and statements that are assembler **directives**,^[5] **macros**,^{[6][1]} and **symbolic labels** of program and **memory locations** are often also supported.

Each assembly language is specific to a particular **computer architecture** and sometimes to an **operating system**.^[7] However, some assembly languages do not provide specific **syntax** for operating system calls, and most assembly languages can be used universally with any



Architecture

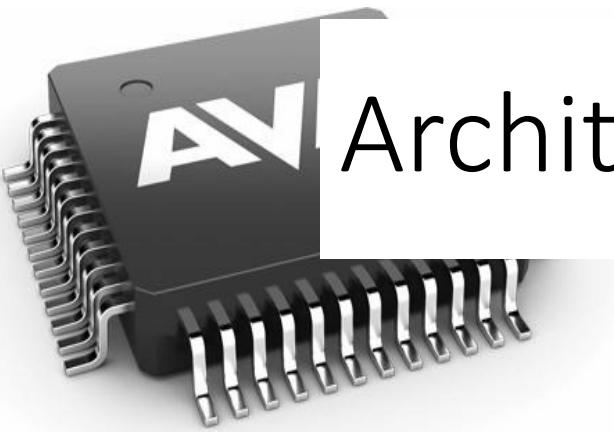
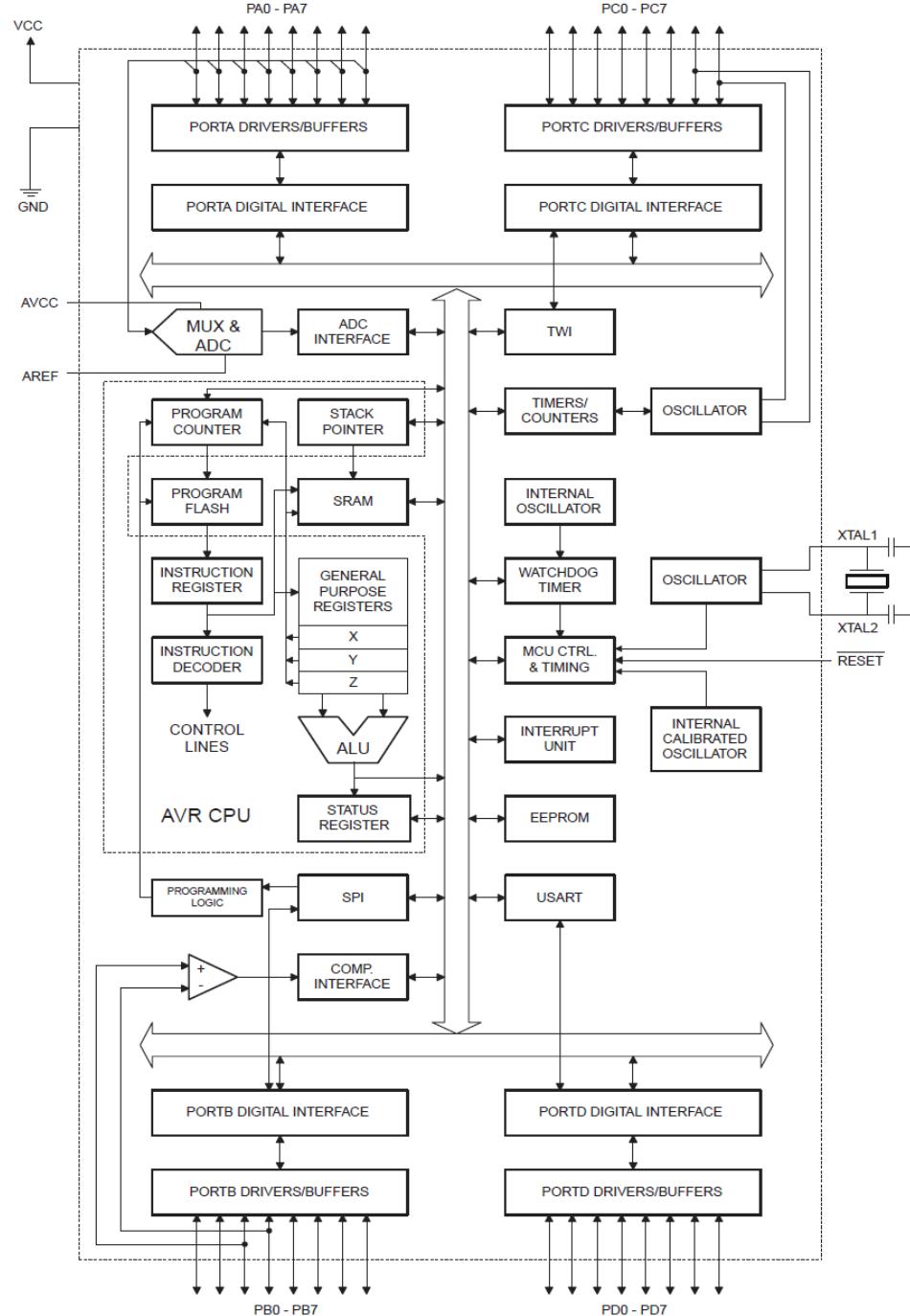


Figure 2. Block Diagram





Different CPUs, different Assembly languages

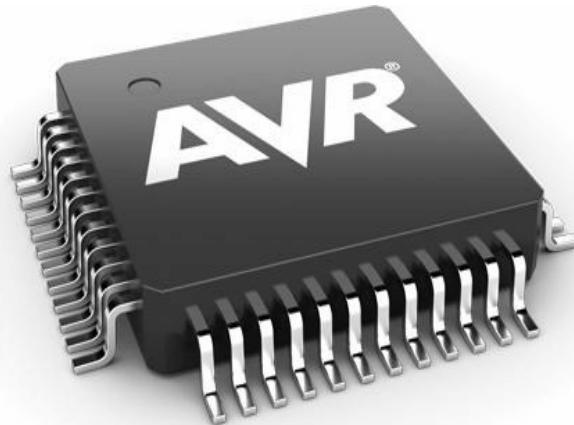
```
Assembly (x86)
1 ; TODO INSERT CONFIG CODE HERE USING CONFIG BITS GENERATOR
2 #INCLUDE <P16F84A.INC>
3 RES_VECT CODE 0x0000          ; processor reset vector
4     GOTO START              ; go to beginning of program
5
6 ; TODO ADD INTERRUPTS HERE IF USED
7
8 CBLOCK 0x0C
9 COUNT1
10 COUNT2
11 ENDC
12
13 MAIN_PROG CODE
14
15 START
16     BSF STATUS, RP0
17     MOVLW 0xFE
18     MOVWF TRISB
19     BCF STATUS, RP0
20
21 MAIN
22     BSF PORTB,0
23     CALL DELAY
24     BCF PORTB,0
25     CALL DELAY
26     GOTO MAIN
27
```



```
//
//asynchronous 8n1 serial transmit
//
#define TX_PIN 6                  //TX I
.global SerialPutChar

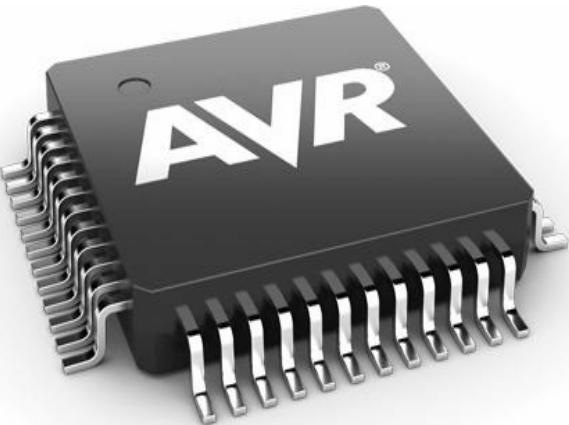
SerialPutChar:
    push r16
    push r17
    ldi r16, 10
    com r24
    sec
    putchar0:
        brcc putchar1
        cbi _SFR_IO_ADDR(PORTA), TX_PIN ;send a '0'
        rjmp putchar2                ;else
    putchar1:
        sbi _SFR_IO_ADDR(PORTA), TX_PIN ;send a '1'
        nop
    putchar2:
        rcall UARTDelay             ;1/2 bit delay +
        rcall UARTDelay             ;1/2 bit delay = 1bit delay
        lsr r24
        dec r16
        brne putchar0               ;If not all bits sent
                                    ;send next
                                    ;else
        pop r17
        pop r16
```





Microchip has purchased Atmel in 2015.
Microcip is thus selling AVR micro controllers.





Documentation.

1. doc2549_ATmega2560.

Most important document.
Includes almost everything about
the controller that you have to
know for programming it.

Example:

Pin lay-out of CPU

Different registers

Memory configuration etc.

Features

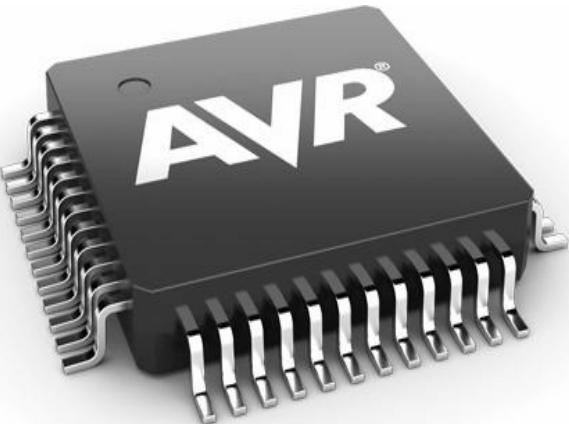
- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High 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
 - 54/62 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)



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

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

Preliminary



Documentation.

1. 8-bit AVR Instruction Set.

Second most important document.
Includes all instructions and
detailed information about them.

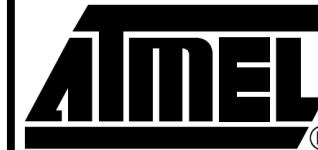
Instruction Set Nomenclature

Status Register (SREG)

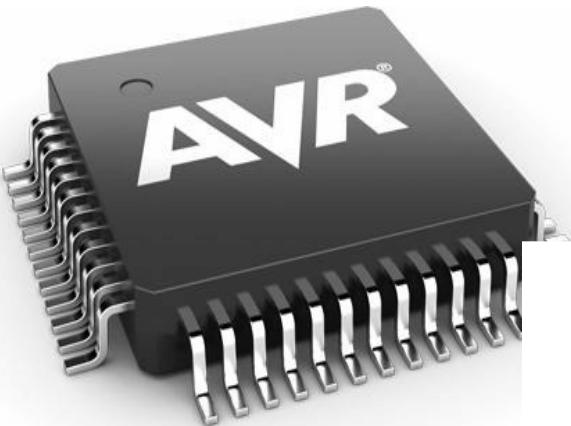
SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	$N \oplus V$, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

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
q:	Displacement for direct addressing (6-bit)



**8-bit AVR®
Instruction Set**

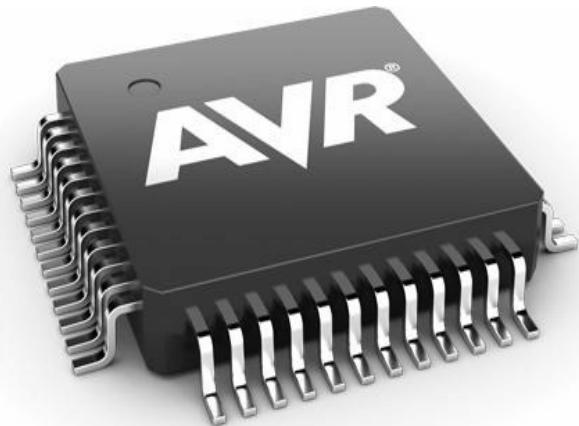


Documentation.

M2560def, include file.

This file includes all definitions and register names. With this file include in a program, we can use names instead absolut addresses.

```
m2560def
***** THIS IS A MACHINE GENERATED FILE - DO NOT EDIT *****
***** Created: 2011-08-25 20:59 ***** Source: ATmega2560.xml *****
*****
;* APPLICATION NOTE FOR THE AVR FAMILY
;*
;* Number          : AVR000
;* File Name       : "m2560def.inc"
;* Title           : Register/Bit Definitions for the ATmega2560
;* Date            : 2011-08-25
;* Version          : 2.35
;* Support E-mail   : avr@atmel.com
;* Target MCU        : ATmega2560
;*
;* DESCRIPTION
;* When including this file in the assembly program file, all I/O register
;* names and I/O register bit names appearing in the data book can be used.
;* In addition, the six registers forming the three data pointers X, Y and
;* Z have been assigned names XL - ZH. Highest RAM address for Internal
;* SRAM is also defined
;*
;* The Register names are represented by their hexadecimal address.
;*
;* The Register Bit names are represented by their bit number (0-7).
;*
;* Please observe the difference in using the bit names with instructions
;* such as "sbr"/"cbr" (set/clear bit in register) and "sbrs"/"sbrc"
;* (skip if bit in register set/cleared). The following example illustrates
;* this:
```



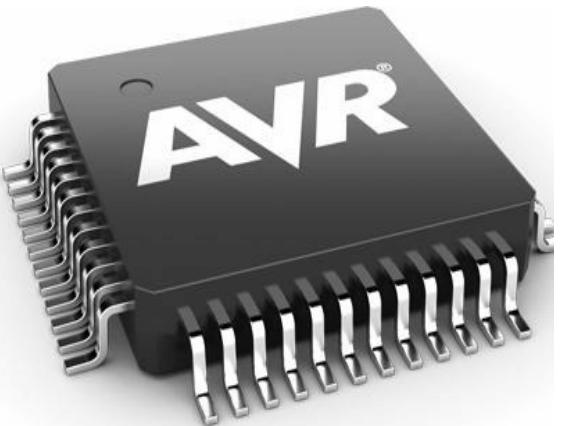
Documentation. m2560def, include file.

Example:

We will use a register which is called DDRA. It is located in this CPU (ATmega2560) on address 0x01. In a different CPU it can be an other address. By including the file m2569def, we can

m2560def

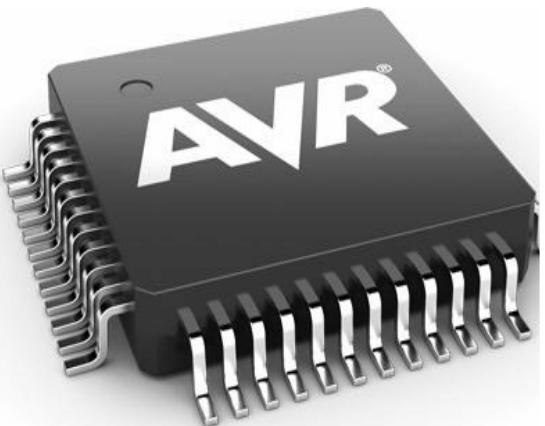
```
.equ    TCNT0    = 0x26
.equ    TCCR0B   = 0x25
.equ    TCCR0A   = 0x24
.equ    GTCCR    = 0x23
.equ    EEARH    = 0x22
.equ    EEARL    = 0x21
.equ    EEDR     = 0x20
.equ    EECR     = 0x1f
.equ    GPIOR0   = 0x1e
.equ    EIMSK    = 0x1d
.equ    EIFR     = 0x1c
.equ    PCIFR    = 0x1b
.equ    TIFR5    = 0x1a
.equ    TIFR4    = 0x19
.equ    TIFR3    = 0x18
.equ    TIFR2    = 0x17
.equ    TIFR1    = 0x16
.equ    TIFR0    = 0x15
.equ    PORTG    = 0x14
.equ    DDRG     = 0x13
.equ    PING     = 0x12
.equ    PORTF    = 0x11
.equ    DDRF     = 0x10
.equ    PINF     = 0x0f
.equ    PORTE    = 0x0e
.equ    DDRE     = 0x0d
.equ    PINE     = 0x0c
.equ    PORTD    = 0x0b
.equ    DDRD     = 0x0a
.EEND
```



Complete Instruction Set Summary

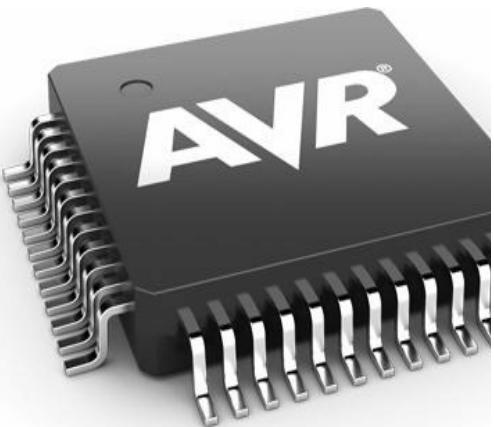
Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
Arithmetic and Logic Instructions					
ADD	Rd, Rr	Add without Carry	$Rd \leftarrow Rd + Rr$	Z,C,N,V,S,H	1
ADC	Rd, Rr	Add with Carry	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,S,H	1
ADIW	Rd, K	Add Immediate to Word	$Rd+1:Rd \leftarrow Rd+1:Rd + K$	Z,C,N,V,S	2 ⁽¹⁾
SUB	Rd, Rr	Subtract without Carry	$Rd \leftarrow Rd - Rr$	Z,C,N,V,S,H	1
SUBI	Rd, K	Subtract Immediate	$Rd \leftarrow Rd - K$	Z,C,N,V,S,H	1
SBC	Rd, Rr	Subtract with Carry	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,S,H	1
SBCI	Rd, K	Subtract Immediate with Carry	$Rd \leftarrow Rd - K - C$	Z,C,N,V,S,H	1
SBIW	Rd, K	Subtract Immediate from Word	$Rd+1:Rd \leftarrow Rd+1:Rd - K$	Z,C,N,V,S	2 ⁽¹⁾
AND	Rd, Rr	Logical AND	$Rd \leftarrow Rd \bullet Rr$	Z,N,V,S	1
ANDI	Rd, K	Logical AND with Immediate	$Rd \leftarrow Rd \bullet K$	Z,N,V,S	1
OR	Rd, Rr	Logical OR	$Rd \leftarrow Rd \vee Rr$	Z,N,V,S	1
ORI	Rd, K	Logical OR with Immediate	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
EOR	Rd, Rr	Exclusive OR	$Rd \leftarrow Rd \oplus Rr$	Z,N,V,S	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	Z,C,N,V,S	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,S,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V,S	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FFh - K)$	Z,N,V,S	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V,S	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V,S	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V,S	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V,S	1



Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
Data Transfer Instructions					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1 ⁽¹⁾
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from data space	Rd ← (k)	None	2 ⁽¹⁾⁽⁴⁾
LD	Rd, X	Load Indirect	Rd ← (X)	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2 ⁽²⁾⁽⁴⁾
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2 ⁽¹⁾⁽⁴⁾
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2 ⁽²⁾⁽⁴⁾
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2 ⁽²⁾⁽⁴⁾
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2 ⁽¹⁾⁽⁴⁾
STS	k, Rr	Store Direct to data space	(k) ← Rd	None	2 ⁽¹⁾⁽⁴⁾
ST	X, Rr	Store Indirect	(X) ← Rr	None	2 ⁽²⁾⁽⁴⁾
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2 ⁽²⁾⁽⁴⁾
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2 ⁽²⁾⁽⁴⁾



MOV – Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

Syntax:

(i) $MOV\ Rd,Rr$

Operands:

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

Program Counter:

$PC \leftarrow PC + 1$

16-bit Opcode:

0010	11rd	dddd	rrrr
------	------	------	------

Status Register (SREG) and Boolean Formula:

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

Example:

```
        mov    r16,r0    ; Copy r0 to r16
        call   check     ; Call subroutine
        ...
check: cpi   r16,$11  ; Compare r16 to $11
```

The decimal system.

The decimal number 2019:

$$2019 = 2 \cdot 1000 + 0 \cdot 100 + 1 \cdot 10 + 9 \cdot 1$$

$$\begin{aligned}2019 &= 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0 = 2 \cdot 1000 + 0 \cdot 100 + 1 \cdot 10 + 9 \cdot 1 = \\&= 2000 + 0 + 10 + 9 = 2019\end{aligned}$$

$$10^0 = ?$$

$$100^0 = ?$$

$$16^0 = ?$$

$$2^0 = ?$$

$$10^0 = 100^0 = 16^0 = 2^0 = 1$$

Each position has a value, which is the base raised to...: ... $10^3, 10^2, 10^1, 10^0$

The decimal system..

Decimal number with fraction:

$$567,318 = 5 \cdot 100 + 6 \cdot 10 + 7 \cdot 1 + 3 \cdot 0,1 + 1 \cdot 0,01 + 8 \cdot 0,001 =$$

$$567,318 = 5 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1} + 1 \cdot 10^{-2} + 8 \cdot 10^{-3} =$$

$$= 500 + 60 + 7 + 0,3 + 0,01 + 0,008 = 567,318$$

The position left of the decimal point has the value the base⁰, in this case 10⁰.

Positional numeral system, each position has a value:

$$\dots 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3} \dots$$

Integer

Fraction

Ev. decimal point

$$10^0 = 100^0 = 16^0 = 2^0 = 1$$

Generellt:

$$a^0 = 1 \text{ for all } a \text{ except 0}$$

Binary numbers.

Example, binary number with 8 bits, integer (= 1 Byte):

1	1	0	1	0	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
			Osv.	$2 \cdot 2 \cdot 2$	$2 \cdot 2$	2	$2^0 = 1$
128	64	32	16	8	4	2	1

Binary number
Positional value, power of 2
Positional value, in decimal

$$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ 1 \cdot 128 + 1 \cdot 64 + 1 \cdot 16 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 215$$

Dvs:

$$1101\ 0111_2 = 215_{10}$$

Radix 2 tells the base is 2,
Binary number system

Radix 10 tells the base is 10,
Decimal system

Binary numbers.

Example, biggest integer value with 8 bits. (= 1 Byte):

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Highest integer with 8 bits.

$$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 =$$

$$1 \cdot 128 + 1 \cdot 64 + 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 255$$

Dvs:

$$1111\ 1111_2 = 255_{10}$$

Binary numbers.

Exampel, smallest integer value with 8 bits. (= 1 Byte):

0	0	0	0	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Smallest integer with 8 bits

$$0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 0$$

Dvs:

$$0000\ 0000_2 = 0_{10}$$

Binary numbers.

Example, a binary number with 8 bits, with fractional part (= 1 Byte), 4 bits integer and 4 bits fraction:

1	1	0	1	0	1	1	1
2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
8	4	2	1	$1/2$	$1/4$	$1/8$	$1/16$

Decimal point, not visible.

Binary number

Positional value, power of 2.

Positional value, decimal.

What is the positional value, now?

$$1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} + 1 \cdot 2^{-4} =$$

$$1 \cdot 8 + 1 \cdot 4 + 1 \cdot 1 + 1 \cdot 0,25 + 1 \cdot 0,125 + 1 \cdot 0,0625 = 13,4375$$

Dvs:

$$1101\ 0111_2 = 13,4375_{10}$$

Please note that the binary number is the same as in previous example, but in this example it's an integer with fraction.
(4 bits integer and 4 bits fraction.)

Compare with:
2019 is not equal to 20,19

Binary numbers, 2-complement form.

Example, binary number with 8 bits, integer (= 1 Byte):

1	1	0	1	0	1	1	1
$-(2^7)$	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-128	64	32	16	8	4	2	1

Binary number
Positional value, power of 2
Positional value, in decimal

$$1 \cdot -(2^7) + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ -1 \cdot 128 + 1 \cdot 64 + 1 \cdot 16 + 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = -41$$

Dvs:

$$1101\ 0111_2 = -41_{10}$$

Radix 2 tells the base is 2,
Binary number system

Radix 10 tells the base is 10,
Decimal system



Two's complement.

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

Inkorgen | LinkedIn pressmeddelande - Expertsvar W Two's complement - Wikipedia

en.wikipedia.org/wiki/Two's_complement

Google

Create account Log in

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.

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

Contents [hide]

1 Potential ambiguities of terminology

Hexadecimala talsystemet.

In the hexadecimal positional system the base is 16. => 16 digits, 0-9, A, B, C, D, E, F.

Decimal:	Hexa-decimal:	Binary:
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010

$1A_{16}$ hexadecimal can be written as:

$$\begin{aligned}1 \cdot 16^1 + A \cdot 16^0 &= \\&= 1 \cdot 16^1 + 10 \cdot 16^0 = \\&= 1 \cdot 16 + 10 \cdot 1 = \\&= 16 + 10 = 26_{10}\end{aligned}$$

$1A9_{16}$ hexadecimal can be written as :

$$\begin{aligned}1 \cdot 16^2 + A \cdot 16^1 + 9 \cdot 16^0 &= \\&= 1 \cdot 16^2 + 10 \cdot 16^1 + 9 \cdot 16^0 = \\&= 1 \cdot 16 \cdot 16 + 10 \cdot 16 + 9 \cdot 1 = \\&= 1 \cdot 256 + 10 \cdot 16 + 9 \cdot 1 = \\&= 256 + 160 + 9 = 425_{10}\end{aligned}$$

$1A9C_{16}$ hexadecimal can be written as :

$$\begin{aligned}1 \cdot 16^3 + A \cdot 16^2 + 9 \cdot 16^1 + C \cdot 16^0 &= \\&= 1 \cdot 16^3 + 10 \cdot 16^2 + 9 \cdot 16^1 + 12 \cdot 16^0 = \\&= 1 \cdot 16 \cdot 16 \cdot 16 + 10 \cdot 16 \cdot 16 + 9 \cdot 16 + 12 \cdot 1 = \\&= 1 \cdot 4096 + 10 \cdot 256 + 9 \cdot 16 + 12 \cdot 1 = \\&= 4096 + 2560 + 144 + 12 = 6812_{10}\end{aligned}$$

Hexadecimala talsystemet.

In the hexadecimal positional system the base is 16. => 16 siffror, 0-9, A, B, C, D, E, F.

Decimalt:	Hexa-decimalt:	Binärt:
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010

1	1	0	1	0	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
8	4	2	1	8	4	2	1

Groups of 4 bits from right.
 $1101 = 8 + 4 + 0 + 1 = 13_{10} = D_{16}$

Groups of 4 bits from right.
 $0111 = 0 + 4 + 2 + 1 = 7_{10} = 7_{16}$

$$Dvs = 1101\ 0111_2 = 215_{10} = D7_{16}$$

The hexadecimal system.

In the hexadecimal positional system the base is 16.

2019 decimal can be written as:

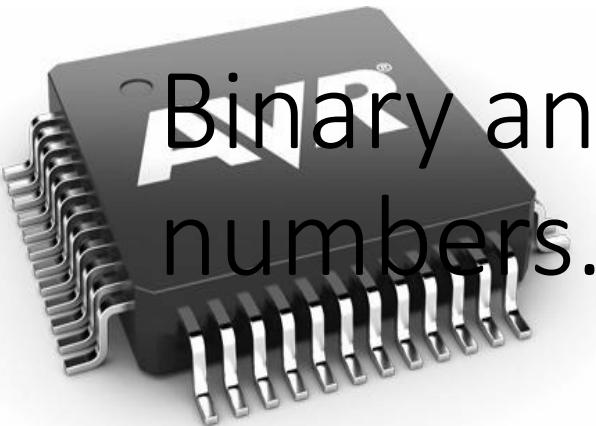
$$2019 = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0 = 2000 + 0 + 10 + 9 = 2019$$

1101 0111₂ binary can be written as:

$$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 215_{10}$$

D7₁₆ hexadecimal can be written as:

$$D \cdot 16^1 + 7 \cdot 16^0 = 13 \cdot 16 + 7 \cdot 1 = 208_{10} + 7 = 215_{10}$$



Binary and hexadecimal numbers.

0001 1110 1101 0000 0010 0001

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

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>



Arkiv Redigera Viga Historik Bokmärken Verktyg Hjälp

Inkorgen | LinkedIn pressmeddelande - Expertsvar Hexadecimal - Wikipedia, ...

en.wikipedia.org/wiki/Hexadecimal

Google Create account Log in

Article Talk Read Edit View history Search

Wiki Loves Monuments: Photograph a monument, help Wikipedia and win!

 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

Hexadecimal

From Wikipedia, the free encyclopedia

In mathematics and computing, **hexadecimal** (also **base 16**, or **hex**) is a **positional numeral system** with a **radix**, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent values zero to nine, and A, B, C, D, E, F (or alternatively a–f) to represent values ten to fifteen. Hexadecimal numerals are widely used by computer systems designers and programmers. In computing, hexadecimal numerals are usually written with a prefix, "0x" (in reference to the abbreviated pronunciation of "hexadecimal"). Alternately, some authors denote hexadecimal values using a suffix or subscript. For example, one could write 0x2AF3 or 2AF3₁₆, depending on the choice of notation.

As an example, the hexadecimal number 2AF3₁₆ can be converted to an equivalent decimal representation. Observe that 2AF3₁₆ is equal to a sum of (2000₁₆ + A00₁₆ + F0₁₆ + 3₁₆), by decomposing the numeral into a series of **place value** terms. Converting each term to decimal, one can further write: (2₁₆ × 16³) + (A₁₆ × 16²) + (F₁₆ × 16¹) + (3₁₆ × 16⁰), (2 × 4096) + (10 × 256) + (15 × 16) + (3 × 1), or 10995.

Each hexadecimal digit represents four binary digits (**bits**), and the primary use of hexadecimal notation is a human-friendly representation of **binary-coded** values in computing and digital electronics. One hexadecimal digit represents a **nibble**, which is half of an **octet** or byte (8 bits). For example, **byte** values can range from 0 to 255 (decimal), but may be more conveniently represented as two hexadecimal digits in the range 00 to FF. Hexadecimal is also commonly used to represent computer **memory addresses**.

Contents [hide]

1 Representation
1.1 Written representation

Numeral systems by culture

Hindu–Arabic origins

Indian (Bengali · Tamil · Telugu)
Eastern Arabic · Western Arabic
Burmese · Khmer · Lao · Mongolian ·
Sinhala · Thai

East Asian

Chinese (Suzhou) · Japanese · Korean ·
Vietnamese
Counting rods

Alphabetic

Abjad · Armenian · Āryabhatā · Cyrillic ·
Ge'ez · Georgian · Greek · Hebrew · Roman

Former

Aegean · Attic · Babylonian · Brahmi ·
Egyptian · Etruscan · Inuit · Kharosthi ·
Mayan · Quipu
Prehistoric

