

expensive
 ← external
 $\text{ AVR } \leftarrow \text{int} \dots, \text{I/O}, \text{RAM}, \text{ROM}, \text{CPU}$
 $\text{ AVR } \leftarrow \text{int} \dots, \text{I/O}, \text{RAM}, \text{ROM}, \text{CPU}$
 $\text{ AVR } \leftarrow \text{int} \leftarrow \text{dedicated} \leftarrow \text{embedded system}$
 $\text{ AVR } \leftarrow \text{int} \leftarrow \text{CPU (microcontroller)} \leftarrow \text{application}$

$\text{ AVR } \leftarrow \text{int} \leftarrow \text{recompile}$ or advanced virtual RISC
 بحسب اجراءات البرمجة خارجية او داخلية
 DIP package - AVR32, AVR16, AVR8 \leftarrow Mega {
 AVR
 Tiny
 Classic
 special purpose }

$\text{ USART, PWM, ADC } \leftarrow \text{int}$ + I/O port - timer - data EEPROM - RAM - ROM
 مثلاً
 \downarrow
 max 64K Flash 8M
 \downarrow
 \downarrow
 1K - 256K
 \downarrow
 مسح الاسم
 EEPROM
 \downarrow
 read GPR AVR to AVR
 $\left\{ \begin{array}{l} \text{GPR} \\ \text{AVR to AVR} \\ \text{I/O mem} \end{array} \right\} \leftarrow \text{internal SRAM}$
 data
 RAM space
 read/write scratch pad
 \downarrow
 مثلاً
 داده های EEPROM
 \downarrow
 86 i 3 \leftarrow I/O pin
 100 i 8 \leftarrow AVR PIN package

com (why) \rightarrow AVR \rightarrow USART + μ C + nib ADC to AVR \rightarrow
 sync receiver transmitter

✓ 16 pins \rightarrow 10 ADC analog input *
 16 I/O pins *

program mem: 4K - 256K
 package: 28 - 100 pin
 rich instructions

ATMega32	Code ROM 32K	data RAM 2K	data EEPROM 1K	I/O pin 32	ADC timer 8	timer 3
		SRAM	MEGA			

TINY \rightarrow program mem: 1K - 8K
 package: 8 - 28 pin

Pin no. and pack
 TQFP44, PDIP40

zigbee LCD controller - CAN controller - USB controller \leftarrow Special purpose
 FPGA

fixed RAM \leftarrow AVR

chapter 2

R0 - R31
 relocatable

w registers \leftarrow GPR
 8 bits

LDI Rd, K
 0 < K < 255
 bit 7ff

{ nothing \rightarrow decimal
 \$, H, 0x \rightarrow hex
 0b \rightarrow binary

error \leftarrow 256 < K

Add Rd, Rr

memory : 1) data
2) code

Add R27
LDI R16, 0x34
LDI R16, 0x16
LDI R19, 0xCD
Add R19, R16

data memory
SFR
GPR \$0000 - \$001F
I/O mem \$0020 - \$005F
internal SRAM \$0060 - \$FFFF
general purpose ram

... , ADC , I/O ports r serial communication , status reg ← I/O mem *

Ext 64 Jladr, Ext mem : wait. ↗

Standard I/O Mem ← Ext 64 Jladr

extended I/O mem ← Ext 64 Jladr / 32 S10pin

Scratchpad ← scratchpad ← internal SRAM *

dataMem = I/O reg + SRAM + GPR

ATMega 32

$$2144 = 64 + \frac{2048}{2^k} + 32$$

0 ≤ d ≤ 31

↑

LDS Rd, K → anywhere

↓
address

\$000 ≤ K ≤ \$FFFF

STS K, Rd

Ext 64K Jladr ← data mem.
Ext 64 ← standard I/O

IN Rd, A → relative address of I/O
 $0 \leq A \leq 63$

OUT A, Rd

I/O, but IN

MOV Rd, Rr → Rd ← Rr
INC Rd → Rd = Rd + 1
DEC Rd → Rd = Rd - 1
SUB Rd, Rr → Rd = Rd - Rr
CLR Rd → Rd = 0

LDS is memory + IN
↓
2 clock

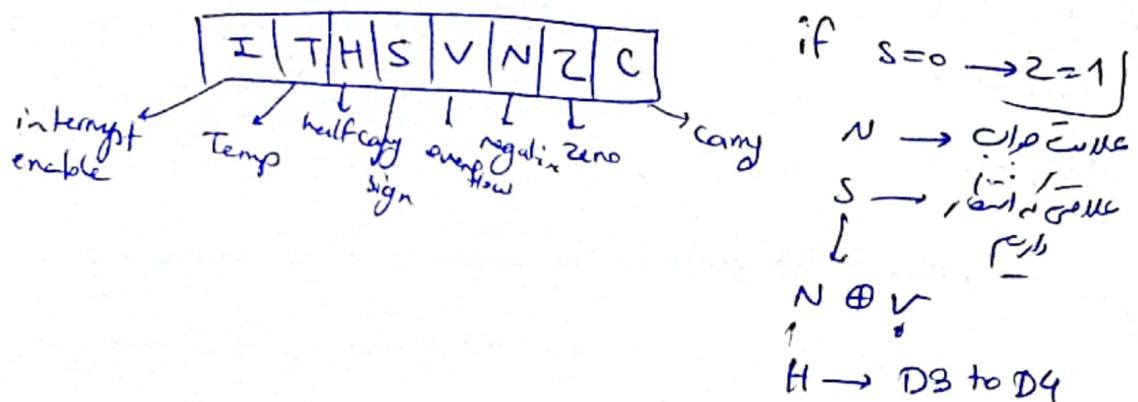
LDS is 5 byte IN
↓
4 byte 2 byte

name ← IN

→ LDS () → name to AVR now IN AT

COM Rd → complement of Rd 0xFFA → 0x55

Status Register → 8-bit I/O reg



العنوان 8 بت يدخل في AND مع العنوان المدخل من الذاكرة لـ ADD من إلستركشن
العنوان 8 بت يدخل في OR مع العنوان المدخل من الذاكرة لـ AVR

'1' = 0x31 → ASCII
'9' = 0x39
string → ""

give directions ← pseudo-instructions ← directives

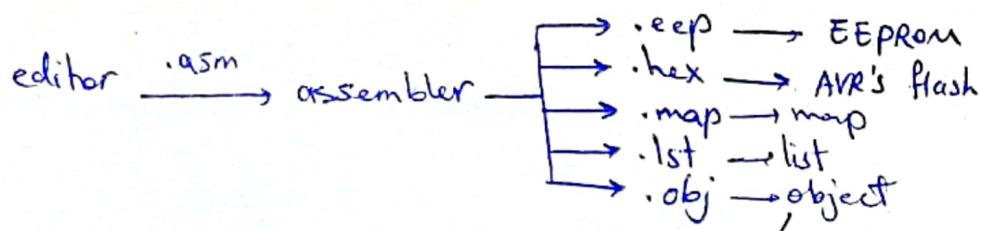
• ORG • DEVICE • EQU

• EQU COUNT = 0x25
• EQU PORTB = 0x1B
LDI R21, COUNT
.INCLUDE "M32DEF.INC"

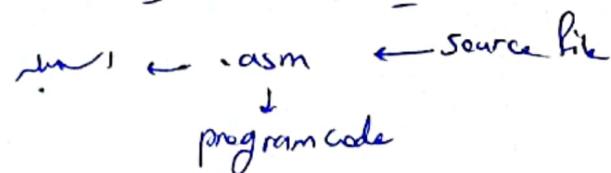
constant value تعرف بـ ثابت

- SET = • EQU
 - address
 - value
 - port
- reassigned later
- origin
- ORG → beginning of the address. code and data
- INCLUDE → add file contents

العنوان يدخل إلى CPU كـ instruction word، و الكود والبيانات كـ opcode، له directive



(DOS) is not ASCII file



ROM file, memory map ← map

Memory map, memory location, optional list
 optional memory location ← list

program Counter

$PC \leftarrow 14$

\$0000-\$03FFF

16K x 2 byte ← 32K ← ATMega32

PC ← 14

8M ← 4M ← 2^{22} ← 22 bits *

Internal ROM location

Rd, K ← 2⁴ bit

LDI →

1110	KKKK	dddd	KKKK
------	------	------	------

all AVR wake up at mem address, 0000 *
 ↓
 PC

K → 8bit 0X EK, d K.
 Rd → 4 bit

ADD →

0000	11	r d	ddddd	rrrr
------	----	-----	-------	------

0, r < 31 → 5 bit

0, d < 31 → 5 bit

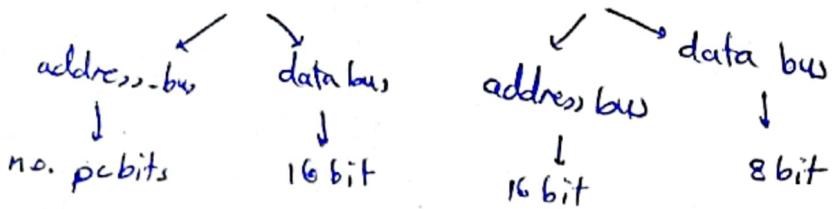
Memory location ← 14 bits *

(R+2) ← 12 bits ← Rom

Memory location ← 2 bytes ← Mem

1,3 X 4 bits ← 12 bits ← AVR Registers

program bus, data bus ← مدارات آدرس و داده



big-endian vs. little-endian

clc

- STS K, Rd

\$06K\$FFFF
16 bit
64 K byte

LDS → STS

- sd\\$31 → 5 bit

high byte ← low byte
mem data

low byte ← low byte
mem data

1001	001r	rrrr	0000
KKKK	--	--	KKKK

16 31 15

- IN Rd, A

sd\\$31
5 bit

1011	0AAd	dddd	AAAA
------	------	------	------

- OUT A, Rd

→ IN Clw

- JMP E → 4-byte

06E9M

10bit opcode
22bit address (RAM)

1001	010k	KKKK	110 K
KKKK	--	--	KKKK

دوره افزاسی مرتب کاسبر : CPU

۱) افزاسی فرستنده clock ← افزاسی معرف توان و تولید ریا

۲) هاردر ورکر arch

۳) تغییر مدار راهنمایی ← استفاده از RISC

RISC ورکر

۱) سازنده دستورات. Ldr, Str و متوجهات ۳ بایت هستند.

۲) سازنده instruction decoder / کنترلر سرور. stack ← حافظه زیرین نیست. برخی RISC ها

۳) تغییر راهنمایی ← حافظه اصلی stack

۱۵) تعداد سیستم دستورات . MUL, SUB, ADD : basic \leftarrow CISC
که عیب: اینها پرورها را نمی توانند.

از طرفی در RISC نیز دستورات بین رده بودند.

Complete Instruction \leftarrow CISC
Complex set computer

RISC بین خطی بین رده بود.

• Ctrl Ovl mem مذکوون شون

clockcycles \leftarrow ۱۳۰ استور clock cycle اجرای شوند . ۷۵٪ در در دستورات (۴۰٪ در دستورات در ۱۵٪ کار سهل اجرای شوند .
که این code scheduling ۱۵٪

Harvard \leftarrow accessing data \leftarrow address bus
accessing opcodes \leftarrow carrying opcodes } compiler
code بین دو دستور (۲۰٪

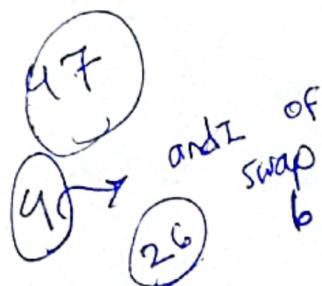
۴۰-۶۰٪ \leftarrow microinstructions در CISC ۱۹٪
۱۰٪ \leftarrow hardware method در RISC

delay \leftarrow ADD reg, mem \leftarrow load/store arch (۲۰٪
X mem, reg بین store, load \leftarrow stall \leftarrow pipeline
X ROM lock \leftarrow ۱۰٪
RISC \rightarrow reduced ...

Add reg1, reg2 \leftarrow GPR \leftarrow data mem \leftarrow EEPROM و AVR

بررسی از پردازش
• opcode \leftarrow pseudo instructions
(2byte) ۱۶bit \leftarrow wide

Chapter 3



$BRNE L \leftarrow DEC$ ← counter ← loop

if zero = 0 ← BRNE

loop \rightarrow ← 255 \downarrow 255 \downarrow

loop \rightarrow ← $\frac{255 \times 255}{65025}$ \downarrow

$Z=1 \leftarrow BREQ$

set the flags according to Rd ← TST Rd

Z, N, S, \dots

$C=0 \leftarrow BRSH$

↓
same or higher

CMP / TST ← \downarrow حاصل من جزء المقارنة

carry INC ← highbyte - lowbyte \downarrow من

• 16-bit flags 84 byte \downarrow short \leftarrow to conditional jump size

$K \downarrow + \frac{PC+1}{16} = \text{target}$

$K [1111 | 01 | KK | KKKK | K000]$

$\rightarrow [-64 < K < 63]$

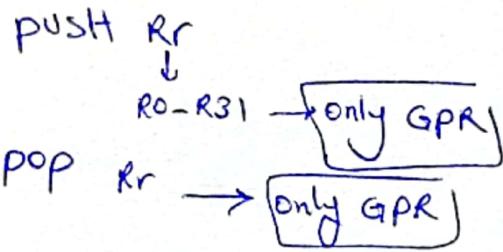
relative address \downarrow 7 ← فار بار \downarrow 3 \downarrow \rightarrow K, to branch ↓
opcode \downarrow 9 signed 2-byte

JMP
4-byte ↓
target ← 22
 $\$00000 - \$3FFFF$
 $0 \leq K \leq 4M$ \downarrow
 $2^{16} = 64K$

$-2048 \leq K \leq 2047$ relative address \downarrow 12

opcode \downarrow 4 C

\downarrow 16 $\ll\ll$ less ROM
 $PC \leftarrow \overline{Z_{reg}} \leftarrow \downarrow 2 \leftarrow IJMP$



LDI R16, HIGH(RAMEND)
 OUT SPH, R16

power up → Sp = 0

Sp → uppermost of RAM

crystal frequency
↳ XTAL1, XTAL2

pipelining

exe + fetch

read + process + WB
operands

branch penalty?

2, 3 or 4 clock cycles

jmp, CALL, RET, all conditional branch

loop in loop → 2bytes ← nop

timer →

Last 4 clock cycles of crystal freq.

CALL DISPLAY

CALL DECAY

RJMP, JMP

3 clock →

4 → CALL, RET

? / 1 255 ↗ 1 256

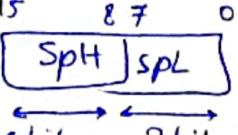
subroutine ← CALL

ret dir + indirect → ↗

4M → opcode → 10 bit → 4 byte
→ 22 bit

subroutine jp → RET

(over) RAM jp over → stack

SP: 
← I/O → i2

Sp = SpH + SpL ← 256 < datamem ~ AVR mem

Sp = SpL ← 256 >

low → high char → 2bytes ← PC ≤ 16
3bytes ← 16 < PC ≤ 24

sp ← sub CALL →

→ sub P + CALL + RET
→ SRAM (obj loc) → stack
← RAM

-2048 ≤ E ≤ 2047 → RCALL

address → 12bit → 2-byte →

ICALL

② → 2-byte →
IJMP (loc)

EI CALL
extended Indirect call) 64K < AVR mem

EIND + Z

I/O

add with carry \leftarrow ADC
multibyte addition

SUBIW \downarrow SUBCI $\frac{SBC}{SUBC}$ SUBI \downarrow SUB
 SUBIW $R_{25}:R_{24}, K$
 $R_{d,1,18}$

(c) \downarrow
 $C \downarrow$ carry

$C=0 \downarrow N=0$
 $C=1 \downarrow N=1$

$R_1 \leftarrow$ high - $R_0 \leftarrow$ low $\leftarrow MUL$

division بجزءين

$10, \text{num} \leftarrow \text{Ans. hex}$ قبل

carry $\leftarrow D_3, D_6 \rightarrow D_7 \quad (1)$ } overflow
 $D_0, D_1, D_2, D_3, D_4, D_5, D_6 \leftarrow$ carry $\leftarrow (2)$

$$V = R_d F \cdot R_{dr} F \cdot \bar{R} F + \bar{R}_d F \cdot \bar{R}_{dr} F \cdot R F$$

$$S = N \oplus V$$

carry $\leftarrow \begin{cases} BRCC \\ BRCS \end{cases}$

BRVC
BRVS

XOR
 $\$FF \leftarrow$ toggle

$2^{\text{num}} \leftarrow \text{NEG}$

Compare $\leftarrow CP$
 Signed \leftarrow BRSH - BRLO

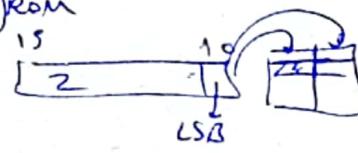
Chapter 6

HIGH (address) → D LDI
 Low (address) → D LDI

LPM R_n, Z → ROM

(program memory) ROM

LSB = 0 → low
 LSB = 1 → high



LD, SBT

Setting bits → SBR / CBR → 0b0010...
 bit

copying → BST / BLD Rd, s

checking → SBRS / SBRC
 skip bit label

Checking flag bit → BRBS S, K
 BRBC

manipulate → BSET s Flag bit set
 BCLR s Flag bit clear

bit addressable AVR 0b_{15..0} ne ✪
 PC, GPR X

BST → T ← 0_{15..0} high 0_{15..0} low
 BCLR 0_{15..0}

EECR ← EEPROM 0_{15..0} OUTN
 EE DR data 0_{15..0} OUTN
 EEAR & H - FEARL 0_{15..0} SBZ CBI

10 ← EEAR ← ATMega320 ✪
 2¹⁰ = 1024

bits of control → EERE

EEWE - EEMWE
 EERIE

EERIE EEMWE EEWE EERE

avoids unwanted write operations

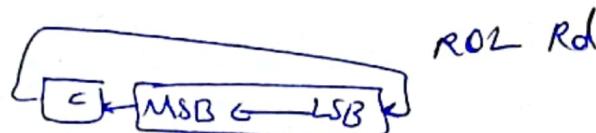
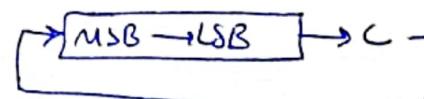
EEWE = 0 → EEAR → EEDR → EEMWE = 1

read

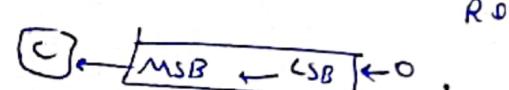
EEMWE = 0 → EEAR → EERE = 1 → read

C,R S=0 ← S Flag } BRGE
 Signed ← } BRLT
 Sel ← }

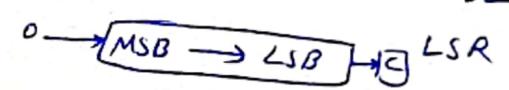
CLC 0 ← ROR Rd



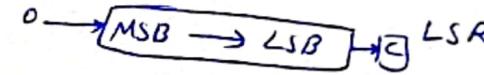
serial port (1)
 one bit at a time (2) } serializing
 ↓



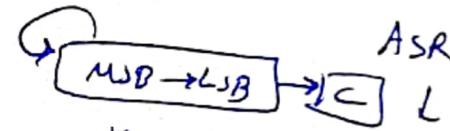
ROR
 ROL



LSL



LSR

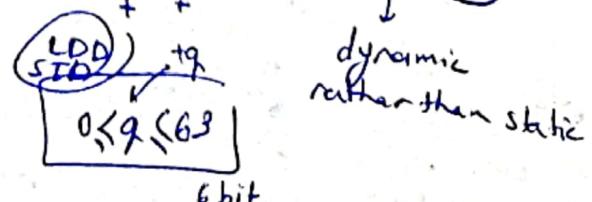


ASR

→ ROL signed
 ← Swap Rd

LSR ←

LDI XL
 LD I XH
 LD Rd, X
 ← register indirect addressing



6 bit

code and
data X → EEPROM

read and write ✓

2's comp \leftarrow

.MACRO name
 @1
---- @0

•END MACRO

• CISTMAC

macro X - .lst

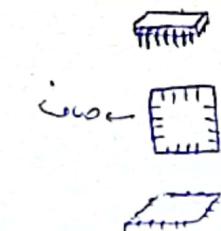
مُعْلِّم - macro EEPROM
ELPM

bit-addressable I/O جستجو 32

bit-addrX ← RAM

SRR

chapter 8



dual in-line package ← DIP ①
 MLF lead frame ← MLF ②
 quad flat package ← QFP ③

ATMega ۳۲ پین

١ ٢ ٣ ATMega32 (32 PIN)
 ۲.۷-۵.۵V : ATMega32L ← +5V ← VCC .
 مدخلات و مخرجات ATMega32L
 مدخلات : A/D → Port A . منبع تزويي برداري ← AVcc .
 مخرجات : ADC ← اعتماد فسفر .

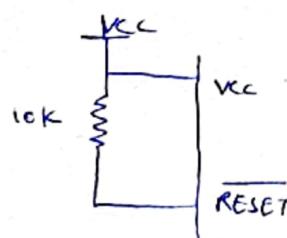
GND, VCC Analog reference pin for ADC ← AREF .

دو سیم برای رسن . برای رسن ۴۰ میلیمتر . معمولاً حین دیسپلی برای رسن .
 رتیوی میکرو ← باهتم که هست که رسن در میکرو که هست .
 دو سیم برای خازن ← XTAL2 , XTAL1 ← دو سیم برای خازن ← ATMega32
 سرعت ۱۶MHz ←

active-low , ((2 سیم ۴۰) ATMega32) = ۰ ← RESET .
 = SRAM جیست هاد .

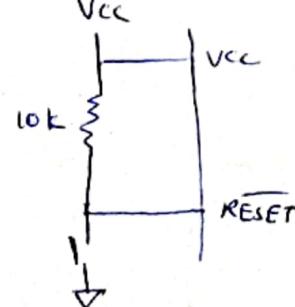
RESET بعده از این پایه از اجری خود را از دیگر CPU
 ۰x000000

① power-on reset



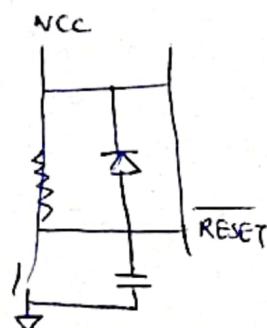
RESET بر وصل درست

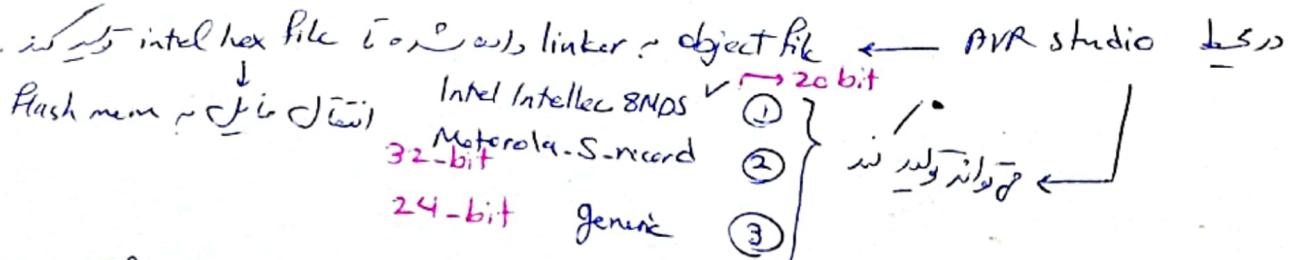
② RESET with momentary switch



③ - Capacitor and diode

حذف برای از از این اثیر را
 پس نهاد خازن رفته خواهد





1M ← extended hex file ←

:BBAAAA TT HHHHHH.....HHHCC ← hex file

load at loc ①

addr ②
loc ③

surrounding ← :

byte 4: A - A - A - A (16) record address ← AAAA
 64KB flash loader loc ← current segment address + 02 ← current segment address

absolute address = record addr + current segaddr < 4
 02 01 00 type ← TT
 02 0000 02 current segaddr ← CC

high byte ← low byte - (current segment address) ← HH HH

checksum ← CC

(Intel hex, CC) 0300 ← byte 4 0030 ← * ←

current seg addr	#0	#1
#1	#2	#3
#2	#4	#5
#3	#6	#7

B₁ B₂ B₃ B₄ ←
 low high

10 + 00 + 08 + ...

برهه کالبی checksum دو رقم صلح می باشد

بسیار بین در تغیرت Carry باشد

* روشِ داده AVR file hex file در درایور

load \rightarrow device burner \rightarrow parallel programming ①

جذب کاربر برای اطلاع از سریال برای حداکثر ۲۵۶ بایت

program \rightarrow این امکان را فراهم می کند. مبتدا فرستید و بعد از اینکه در

SPI (serial peripheral interface) در داده ایجاد شود. داده

JTAG \rightarrow ATMega \rightarrow هارد دیسک اسپرت

VCC, GND, Reset \rightarrow AVRISP

و دلیل این این است که نیازی نیست (خط خود را جدا نکنید) \rightarrow JTAG

boot loader \rightarrow سیستم غیر قرار گرفته شود.

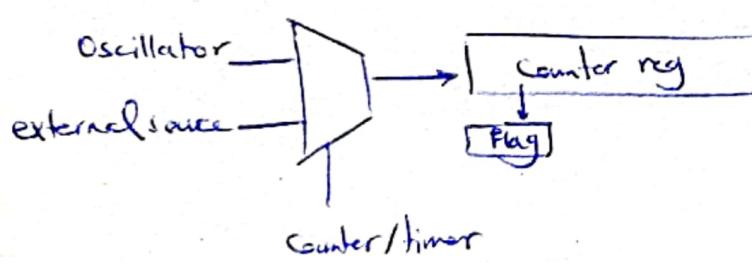
و این این این است که برد کاربر این را load کند. از آن طور نشانه شود.

پورت سریال، پورت شبکه Network & USB، CAN

و تواند JTAG پر دیگر کردن (ستفاده شود).

بعض اصلی این است که پورت ارتباطی و مختار زنگ سیزده را دارد.

* باید متن اینکه دسته مورد استفاده قرار گیرد program شود (تسلط بدل از دریں میل).



chapter 9
در تابع زیر از این ایجاد شود

روشن کردن ایجاد تا خود

۱) صفر کنید و سپس ۱ بین عدد مفہوم پیدا کنید counter ①

۲) عدد مفہوم را در آن کام کنید و سپس کنید overflow ②

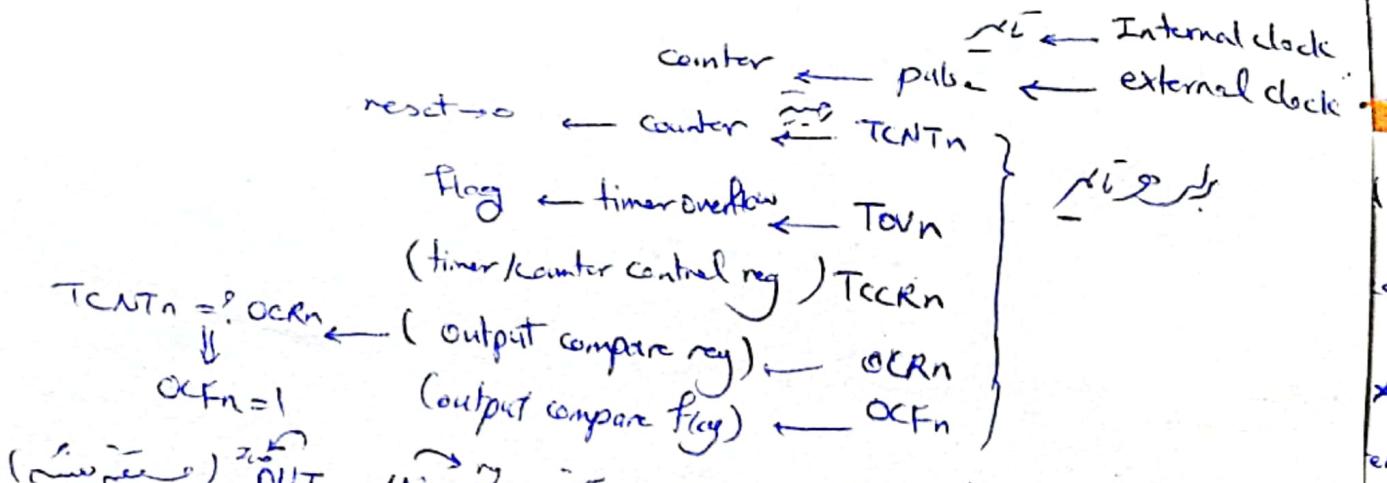
(timer2, timer0) = 0, 1, ..., 5 \rightarrow AVR

8 \rightarrow 2 \rightarrow counter timer

(timer1) = 16 \rightarrow 1 \rightarrow سیم تابیرها

16 \rightarrow 2 \rightarrow 16

③



CLK
CLK/8
CLK/64
CLK/256
CLK/1024
F_e
re

CLK
CLK/8
CLK/32
CLK/64
CLK/128
CLK/256
CLK/1024

TCCR0 (timer/counter control register)

7	6	5	4	3	2	1	0
F _{OC0}	WGM ₀₀	C _{OCR1}	C _{OCR0}	WGM ₀₁	C _{S02}	C _{S01}	C _{S00}
w	RW	RW	RW	RW	RW	RW	RW

F_{OC0} → force compare match, write-only,

WGM₀₀, WGM₀₁

- | | |
|----------------|-----------------------------|
| D ₆ | D ₃ |
| 0 | 0 → Normal |
| 0 | 1 → CTC (clear timer on CM) |
| 1 | 0 → PWM, phase correct |
| 1 | 1 → Fast PWM |

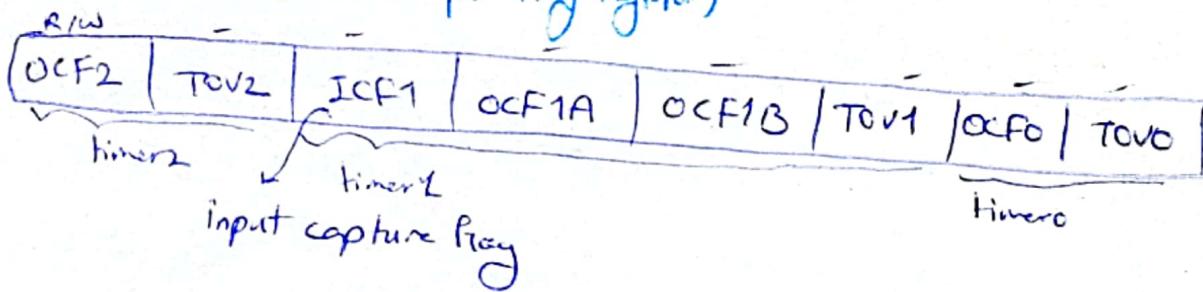
COM 01:00

Compare output mode
control waveform generator

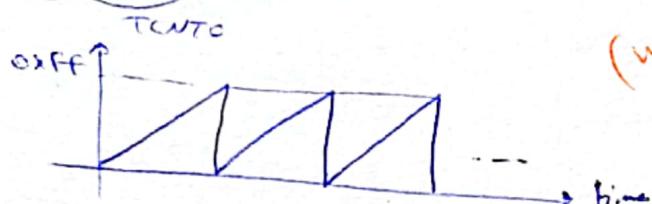
C_{S02}: 00

- | | | |
|----------------|----------------|------------------------------|
| D ₂ | D ₁ | D ₀ |
| 0 | 0 | 0 → No Jack (stop) |
| 0 | 0 | 1 → CLK (No pres) |
| 0 | 1 | 0 → CLK/8 |
| 0 | 1 | 1 → CLK/64 |
| 1 | 0 | 0 → CLK/256 |
| 1 | 0 | 1 → CLK/1024 |
| 1 | 1 | 0 → external clock (Palling) |
| 1 | 1 | 1 → n - (rising) |

TIFR (timer/counter interrupt flag register)



LDI R20, 0x01
OUT TIFR, R20



(WGM00..1 = 00) Normal Mode

: Normal Mode, timer0 oscillates

$$TCNT0 = \text{initial} \quad ①$$

$$TCCR0 \quad ②$$

$$\text{LDI R20, } \text{ex. } 00 \quad ③$$

$$\text{OUT TCCR0, R20} \quad ④$$

$$\text{SBI DDRB, 5} \quad ⑤$$

$$\text{SBR5 R20, TOV0} \quad ⑥$$

$$\text{TOV0} \quad ⑦$$

$$\text{LDI R20, } (1 \ll \text{TOV0}) \quad ⑧$$

$$\therefore 22 \text{ ms} + \frac{98}{194}$$

$$\frac{256}{62} = \frac{1}{194}$$

$$① FF - XX+1$$

$$② 256 - \text{decimal}$$

$$TCNT0 = XX \quad ⑤$$

$$\text{hex} \quad ④$$

$$256 - n \quad ③$$

$$\frac{\text{timertime}}{\text{Tclock}} \quad ②$$

$$T_{clock} = \frac{1}{f_{timer}} \quad ①$$

(clear time on em) CTC Mode

$$TCNT0 \rightarrow OCR0 \rightarrow TCCR0 \rightarrow OCFO$$

$$OCR0 \leftrightarrow OCFO$$

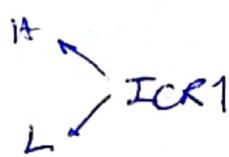
roll-over \leftarrow if $TCCR0 > OCR0$

Timer 2

Tosc2, Tosc1 \approx 32.768 kHz
 انتظامیت عدد Counter
 در زمانی که راکت AS2 می‌گردید
 تریکل - و پس CS02:00

FOC2	WGM20	com21	com20	WGM21	CS22	CS21	CS20
w	RW	-	-	-	-	-	-

NOCK
 CLK
 /8
 /32
 /64
 /128
 /256
 /1024



Timer 1

OCR1AH
OCR1AL
OCR1A

TCCR1A

com1A1	com1A0	com1B1	com1B0	FOC1A	FOC1B	WGM11	WGM10
R				chanA		chanB	
Compare output channel A				channel B			

TCCR1B

ICNC1	ICES1	-	WGM13	WGM12	D ₃	D ₂	D ₁	D ₀
R								

0100 \rightarrow CTC
1100 \rightarrow CTC

Same as before
Timer 0

TCCR1A \rightarrow 0
TCCR1B \rightarrow 8

\$FFFF \leftarrow Normal
0100 \leftarrow CTC

FFFF - x + 1

این پسچشی +

LDI R20, HIGH(addr)
 OUT TCNT1H, R20
 LDI R20, LOW(addr)
 OUT TCNT1L, R20

LDI R16, 0x15

Temporary

OUT TCNT1H, R16

LDI R16, 0xFF

OUT TCNT1L,R16 \sim TCNT1H, R16 Temp

العنوان ينبع من high low

TCNT1H, TCNT1L \leftarrow عنوان الموضع

عنوان الموضع *

(handler)

ISR (interrupt service routine) \leftarrow ISR
بروتوكول دفع ادرس بـ ISR

Chapter 10

interrupt vector table \leftarrow IVT
بروتوكول دفع ادرس بـ IVT

INT2 \leftarrow 006 INT1 \leftarrow 004 INT0 \leftarrow 002

مراحل فحص AVR : interrupt
(pc) stack دسترسی ، خطا ، دسترسی برای ①

ISR (interrupt routine) \leftarrow IVT ②

اجل ارجاع RETI \leftarrow ISR ③

بر اساس نتائج مبلغ برگردان ④

: AVR \rightarrow interrupt

CTC Compare match
overflow ~~and~~ \leftarrow لامپ ①

pins PB2, PP3 (portD.2) PDD0 ① : بـ ISR بـ interrupt ②

.INT2, INT1, INT0

USART بـ ISR بـ interrupt ③

SPI (Serial Peripheral Interface) ADC ④

ادسنس انتقال جمله JMP بـ 4 بت (word) 2word - ISR بـ ادرس انتقال

(masked interrupt) SREG مُنْهَى لـ interrupt. reset flag تردد فل. flag من مُسْخَنَة. بิต 7 (D7) جُمِعَتْ سُفَلَ فل / مُنْهَى لـ كُلُّ وقتِها است.

D7	T	H	S	V	N	Z	C
----	---	---	---	---	---	---	---

* طلاق فل درون و مدن:

(SEI) ١ → SREG, I بٰت ①

اب بٰجِي جو هُو جُمِعَتْ بـ enable من بٰت، TIMSK جُمِعَتْ ②

①	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
		timer1 capture interrupt			↓ output compare match interrupt enable			↓ overflow interrupt enable	

SBR
CBR

CBR R20, ١ < TOIE0
OUT TIMSK, R20

TIFR, TOV1 On ١ لـ ISR ، C1E → SREG، I ، T1I ١ لـ جُمِعَتْ ← TOIE1

TIFR, OCIE1A ١ لـ ISR ، C1E → SREG، I ، T1I ١ لـ جُمِعَتْ ← OCIE1A



LDI R16, ١ < TOIE0
OUT TIMSK, R16

IN → PIN
OUT → PORT

14-10 نمودار

دباره ISR ای
بر مدل میگیرد

در آرس 200 پر گردید

خودکاراً، I، 6 RETI *

AVR خود ISR ؟ On ١ لـ جُمِعَتْ تOV0 لـ ISR و

و دلیل، J - RETI، دلیل کرد و با PC - stack is در درج ← RET، RETE زیر

23, 22 نمودار

(push) مدخلة edge-triggered
• low - Machine cycle 5 pin ← level-triggered

* دوادیت interrupt

i) INTO ١٢ - زیر اجرا شود. پس از آن FSR لذوق خود را فروخته است.

• INT1 \$0006, INT2 \$0001, \$0002 INTO آرین نمایند → دوادیت سینه

و زیر اجرا شود. در حال اجرا ISR interrupt

RETI → دفعه از این قسم است → ISR و درین قسم سریع است

PUSH R20 resource conflict ۱

IN R20, PIND ۲ context switch

POP R20

RETI

• CPU نسبت ۱۰٪

۳ save stack, SREG

PUSH R20

IN R20, SREG

PUSH R20

POP R20

CVT SREG, R20

POP R20

RETI

interrupt latency *

task CPU بین فرآوری و CPU بین فرآوری

لجزی .