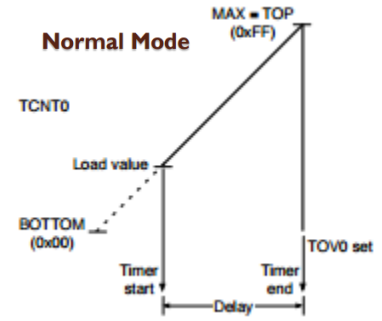
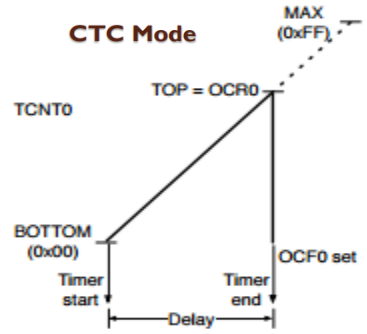




$$Delay_{Normal} = \frac{(MAX - value) \cdot prescale}{clk_{I/O}}$$

$$Delay_{CTC} = \frac{TOP \cdot prescale}{clk_{I/O}}$$

1 sec using the 8-bit Timer/Counter0 16 MHz operating under Normal mode.

```
.include "m128def.inc"
.def mpr = r16
.def counter = r17
…
.ORG    $0000
    RJMP Initialize
.ORG    $0020           ; Timer/Counter0 overflow interrupt vector
    RCALL Reload_counter
    RETI
.ORG    $0046           ; End of interrupt vectors

Initialize:
    LDI  mpr, 0b00000100 ; Enable interrupt on Timer/Counter0 overflow
    OUT  TIMSK, mpr
    SEI                  ; Enable global interrupt
    LDI  mpr, 0b00000111 ; Set prescalar to 1024
    OUT  TCCR0, mpr      ;
    LDI  mpr, 99         ; Load the value for delay
    OUT  TCNT0, mpr      ;
    LDI  counter, 100    ; Set timer
LOOP:
    CPI  counter, 0      ; Repeat for 100 times
    BRNE LOOP
    …

Reload_counter:
    PUSH mpr
    LDI  mpr, 16         ;
    OUT  TCNT0, mpr      ; Load the value for delay
    DEC  counter
    POP  mpr
    RET
```

```
initUSART0:
    ; Port E set up – pin1 output
    ldi  mpr, 0b00000010      ; configure USART0
    out  DDRE, mpr            ; pin direction output
    ; Set Baud rate
    ldi  mpr, 51              ; set baud rate to 19,200 with f = 16 MHz
    out  UBRR0L, mpr          ; UBRR0H already initialized to $00
    ; Enable transmitter
    ldi  mpr, (1<<TXEN0)
    out  UCSR0B, mpr
    ; Set asynchronous mode
    ldi  mpr, (0<<UMSEL0)
    sts  UCSR0C, mpr          ; UCSR0C in extended I/O space, use sts
    ; Set frame format
    ldi  mpr, (0<<USBS0|0<<UPM01|0<<UPM00|0<<UCSZ02|1<<UCSZ01|1<<UCSZ00)
    sts  UCSR0C, mpr          ; UCSR0C in extended I/O space, use sts
    ; Disable interrupt
    ldi  mpr, (0<<TXCIE0)     ; Disable interrupt
    out  UCSR0B, mpr          ;
```

- 8 data bits, 1 stop bit, and no parity
- 19, 200 baud rate
- Transmitter enabled
- Normal asynchronous mode operation
- No interrupts

ISCn1:0 External Interrupt n Sense Control Bits
00 - Low level generates an interrupt request.
01 - Reserved (for ISC3-0)
Any logical change on generates an interrupt request (ISC7-4)
10 - Falling edge generates an interrupt request.
11 - Rising edge generates an interrupt request.

```
;USART1
    ldi mpr, (1<<UDRE1)
    sts UCSR1A, mpr ; Double USART transmit speed
    ;Set baudrate at 2400bps
    ldi mpr, high(832) ; Load high byte of 0x0340
    sts UBRR1H, mpr ; UBRR0H in extended I/O space so sts
    ldi mpr, low(832) ; Load low byte of 0x0340
    sts UBRR1L, mpr
    ;Enable receiver and enable receive interrupts
    ldi mpr, (1<<RXEN1 | 1<<RXCIE1)
    sts UCSR1B, mpr
    ;Set frame format: 8data bits, 2 stop bit
    ldi mpr, (0<<UMSEL1 | 1<<USBS1 | 1<<UCSZ11 | 1<<UCSZ10)
    sts UCSR1C, mpr ; set to Asynchronous mode, 2 stop bits, 8 bit cha
```

External Interrupt Flag Register (EIFR)

7	6	5	4	3	2	1	0	
INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	

INTFn = 1 Triggers interrupt request

External Interrupt Mask register (EIMSk)

7	6	5	4	3	2	1	0	
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	

INTn = 1 Enables interrupt

External Interrupt Control Register A (EICRA)

7	6	5	4	3	2	1	0	
ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	

External Interrupt Control Register B (EICRA)

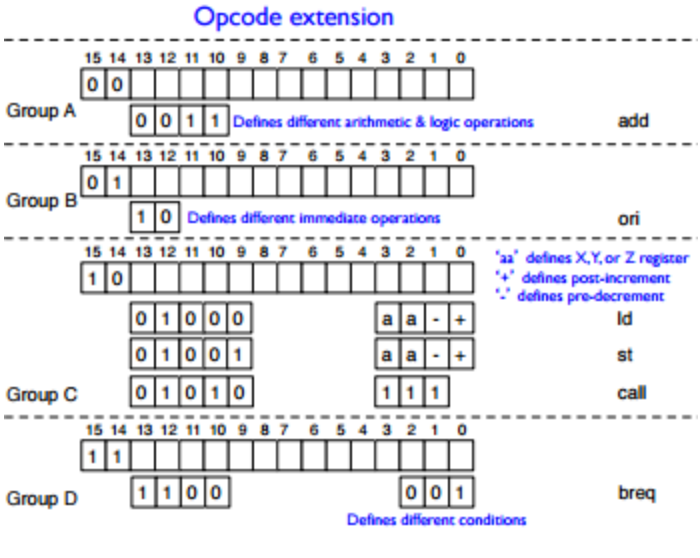
7	6	5	4	3	2	1	0	
ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	

16-bit Timer/Counter1 with the system clock frequency of 16 MHz using the Normal mode.

$$value = 65535 - (1\ ms/(\ prescale \times 62.5\ ns)) = 65535 - (16000/prescale)$$

```
; AVR assembly code – Wait 1 ms
WAIT_1msec:
    LDI  mpr, 0b00000001 ;
    OUT  TCCR1B, mpr     ; Set prescalar to 1
    LDI  mpr, low(49535) ;
    OUT  TCNT1L, mpr     ; (Re)load the low byte of value for delay
    LDI  mpr, high(49535)    ;
    OUT  TCNT1H, mpr     ; (Re)load the high byte of value for delay
LOOP:
    IN   mpr, TIFR       ; Read in TOV1
    ANDI mpr, 0b00000100 ; Check if its set
    BREQ LOOP            ; Loop if TOV1 not set
    LDI  mpr, 0b00000100 ; Reset TOV1
    OUT  TIFR, mpr       ; Note – write 1 to reset
    RET
```

```
WAIT_1sec:
    LDI R30, low(1000) ; Load low byte of count = 1000
    LDI R31, high(1000) ; Load high byte of count = 1000
    Loopy:
        RCALL WAIT_1msec ; Call 1 msec delay subroutine
        SBIW R30, 1 ; Decrement counter
        BREQ Check_Upper
        RJMP Loopy
    Check_Upper:
        BREQ Skip
        RJMP Loopy
    Skip: nop
    ret
```




```
USART_Transmit:

    ;sbis UCSR1A, UDRE1 ; Loop until UDR0 is empty
    ldi mpr, UCSR1A
    cpi mpr, 0b00100000     ;checks 5th bit, UDRE1 if it is set
    brne transmit
    rjmp USART_Transmit

    transmit: nop
    sts UDR1, loadcmd ; Move data to Transmit Data Buffer
    ret
```

RET

EX1: SP ← SP + 1

EX2: PCh ← M[SP], SP ← SP + 1

EX3: PCl ← M[SP]

RAL Output	RET		
	EX1	EX2	EX3
wA	x	x	x
wB	x	x	x
rA	x	x	x
rB	x	x	x

Control Signals	IF	RET		
		EX1	EX2	EX3
MJ	0	x	x	x
MK	0	x	x	x
ML	0	x	x	x
IR_en	1	0	0	x
PC_en	1	0	0	0
PCh_en	0	0	1	0
PCl_en	0	0	0	1
NPC_en	1	x	x	x
SP_en	0	1	1	0
DEMUX	x	x	1	0
MA	x	x	x	x
MB	x	x	x	x
ALU_f	xxxx	xxxx	xxxx	xxxx
MC	xx	xx	xx	xx
RF_wA	0	0	0	0
RF_wB	0	0	0	0
MD	x	x	x	x
ME	x	x	0	0
DM_r	x	x	1	1
DM_w	0	0	0	0
MF	x	x	x	x
MG	x	x	x	x
Adder_f	xx	xx	xx	xx
Inc_Dec	x	0	0	x
MH	x	x	x	x
MI	x	x	x	x

LPM instruction.

EX1: PMAR ← Zh:Zl

EX2: MDR ← M[PMAR]

EX3: R0 ← MDR

RAL Output	LPM		
	EX1	EX2	EX3
wA	x	x	x
wB	x	x	R0
rA	Zh	x	x
rB	Zl	x	x

Control Signals	IF	LPM		
		EX1	EX2	EX3
MJ	0	x	x	x
MK	0	x	x	x
ML	0	x	1	x
IR_en	1	0	0	x
PC_en	1	0	0	0
PCh_en	0	0	0	0
PCl_en	0	0	0	0
NPC_en	1	x	x	x
SP_en	0	0	0	0
DEMUX	x	x	x	x
MA	x	x	x	x
MB	x	x	x	x
ALU_f	xxxx	xxxx	xxxx	xxx
MC	xx	xx	xx	10
RF_wA	0	0	0	0
RF_wB	0	0	0	1
MD	x	x	x	x
ME	x	x	x	x
DM_r	x	x	x	x
DM_w	0	0	0	0
MF	x	x	x	x
MG	x	1	x	x
Adder_f	xx	11	xx	xx
Inc_Dec	x	x	x	x
MH	x	x	x	x
MI	x	x	x	x

RCALL instruction.

EX1: M[SP] ← RARl, SP ← SP - 1

EX2: M[SP] ← RARh, SP ← SP - 1, PC ← NPC + se k

RAL Output	RCALL	
	EX1	EX2
wA	x	x
wB	x	x
rA	x	x
rB	x	x

Control Signals	IF	RCALL	
		EX1	EX2
MJ	0	x	1
MK	0	x	x
ML	0	x	x
IR_en	1	0	x
PC_en	1	x	1
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	0	x
SP_en	0	1	1
DEMUX	x	x	x
MA	x	x	x
MB	x	x	x
ALU_f	xxxx	xxxx	xxxx
MC	xx	01	01
RF_wA	0	0	0
RF_wB	0	0	0
MD	x	0	0
ME	x	0	0
DM_r	x	0	0
DM_w	0	1	1
MF	x	x	0
MG	x	x	0
Adder_f	xx	xx	00
Inc_Dec	x	1	1
MH	x	x	x
MI	x	0	1

LD Rd,Y+ instruction

EX1: DMAR ← Yh:YL, Yh:Yl ← Yh:Yl + 1

EX2: Rd ← M[DMAR]

RAL Output	LD Rd, Y+	
	EX1	EX2
wA	Yh	x
wB	Yl	Rd
rA	Yh	x
rB	Yl	x

Control Signals	IF	LD Rd, Y+	
		EX1	EX2
MJ	0	xx	xx
MK	0	x	x
ML	0	x	x
IR_en	1	0	x
PC_en	1	0	0
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	x	x
SP_en	0	0	0
DEMUX	x	x	x
MA	x	x	x
MB	x	x	1
ALU_f	xxxx	xxxx	xxxx
MC	xx	01	01
RF_wA	0	1	0
RF_wB	0	1	1
MD	x	x	x
ME	x	x	1
DM_r	x	x	1
DM_w	0	0	0
MF	x	x	x
MG	x	1	x
Adder_f	xx	01	xx
Inc_Dec	x	x	x
MH	x	0	x
MI	x	x	x

CPI Rd,K *(Compare Register with Immediate)*

EX: Rd - K

RAL Output	CPI EX
wA	x
wB	x
rA	Rd
rB	x

Control Signals	IF	CPI EX
MJ	00	xx
MK	0	x
ML	0	x
IR_en	1	x
PC_en	1	0
PCh_en	0	0
PCl_en	0	0
NPC_en	1	x
SP_en	0	0
DEMUX	x	x
MA	x	0
MB	x	x
ALU_f	xxxx	0010
MC	xx	xx
RF_wA	0	0
RF_wB	0	0
MD	x	x
ME	x	x
DM_r	x	x
DM_w	0	0
MF	x	x
MG	x	x
Adder_f	xx	xx
Inc_Dec	x	x
MH	x	x
MI	x	x

RAL Output	MOVW EX
wA	Rd+1
wB	Rd
rA	Rr+1
rB	Rr

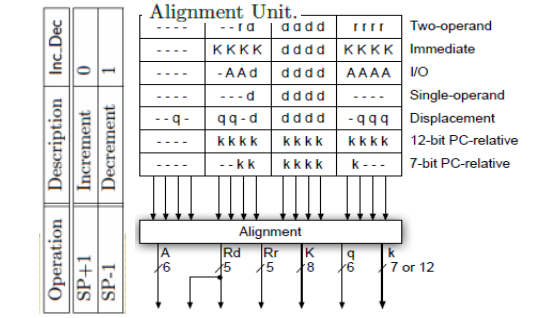
Control Signals	IF	MOVW EX
MJ	0	x
MK	0	x
ML	0	x
IR_en	1	x
PC_en	1	0
PCh_en	0	0
PCl_en	0	0
NPC_en	1	x
SP_en	0	0
DEMUX	x	x
MA	x	x
MB	x	x
ALU_f	xxxx	xxxx
MC	xx	01
RF_wA	0	1
RF_wB	0	1
MD	x	x
ME	x	x
DM_r	x	x
DM_w	0	0
MF	x	x
MG	x	1
Adder_f	xx	11
Inc_Dec	x	x
MH	x	x
MI	x	x

RAL Output	LD Rd, -X	
	EX1	EX2
wA	Xh	x
wB	Xl	Rd
rA	Xh	x
rB	Xl	x

Control Signals	IF	LD Rd, -X	
		EX1	EX2
MJ	0	x	x
MK	0	x	x
ML	0	x	x
IR_en	1	0	x
PC_en	1	0	0
PCh_en	0	0	0
PCl_en	0	0	0
NPC_en	1	x	x
SP_en	0	0	0
DEMUX	x	x	x
MA	x	x	x
MB	x	x	1
ALU_f	xxxx	xxxx	xxxx
MC	xx	01	00
RF_wA	0	1	0
RF_wB	0	1	1
MD	x	x	x
ME	x	x	1
DM_r	x	x	1
DM_w	0	0	0
MF	x	x	x
MG	x	1	x
Adder_f	xx	10	xx
Inc_Dec	x	x	x
MH	x	1	x
MI	x	x	x

Load and Store Instructions				
Stage	Micro-operations			
	Normal	Displacement	Pre-Decrement	Post-Increment
EX1	DMAR ← AR	DMAR ← AR+q	DMAR ← AR-1, AR ← AR -1	DMAR ← AR, AR ← AR +1
EX2	Loads		Stores	
	Rd ← M[DMAR]		M[DMAR] ← Rr	

Alignment Unit.




```
INIT:   ldi   mpr, 0b00001111 (1)  ; Set DDRA to control engine
        out   DDRA, mpr       (2)  ;
        ldi   mpr, 0b00000000 (3)  ; Set DDRD to detect whiskers
        out   DDRD, mpr       (4)  ;
        ldi   mpr, 0b00000011 (5)  ; Enable pull-up resisters for L/R whiskers
        out   PORTD, mpr      (6)  ;
        ldi   mpr, 0b00001010 (7)  ; Set EICR to detect on falling edge
        sts   EICRA, mpr      (8)  ;
        ldi   mpr, 0b00000011 (9)  ; Set EIMSK
        out   EIMSK, mpr      (10) ;
        sei                        ; Turn on interrupts
```

```
.ORG    $0000
    RJMP Initialize
.ORG    $0020           ; Timer/Counter0 overflow interrupt vector
    RCALL Reload_counter
    RETI
.ORG    $0046           ; End of interrupt vectors

Initialize:
    LDI  mpr, 0b00000100 ; Enable interrupt on Timer/Counter0 overflow
    OUT  TIMSK, mpr
    SEI                  ; Enable global interrupt
    LDI  mpr, 0b00000111 ; Set prescalar to 1024
    OUT  TCCR0, mpr      ;
    LDI  mpr, 99         ; Load the value for delay
    OUT  TCNT0, mpr      ;
    LDI  counter, 100    ; Set timer
LOOP:
    CPI  counter, 0      ; Repeat for 100 times
    BRNE LOOP
    …

Reload_counter:
    PUSH mpr
    LDI  mpr, 16         ;
    OUT  TCNT0, mpr      ; Load the value for delay
    DEC  counter
    POP  mpr
    RET
```

```
.def mpr = r16                  ; Multi-purpose register
.def count = r17                ; Assume R17 is initially 0
.ORG $0000
START: RJMP INIT
.ORG $0002
       RCALL ISR
       RETI
INIT:  LDI   mpr, 0b00000011   ; Sets Input Sense Control for pin INT0
       STS   EICRA, mpr        ; to detect an interrupt on a rising edge
       LDI   mpr, 0b00000001   ; Enables interrupt for pin INT0
       OUT   EIMSK, mpr        ;
       LDI   mpr, $00          ; Set Port A Direction Register for input
  OUT  DDRA, mpr               ;
       SEI                     ; Turn on interrupts
       LDI   XH, high(CTR)
       LDI   XL, low(CTR)
       LDI   YH, high(DATA)
       LDI   YL, low(DATA)
WAIT:  RJMP WAIT
.ORG   0x100F
ISR:   IN    mpr, PINA
       ST    Y+, mpr
       INC   count
       ST    X, count
       RET
.DSEG
CTR:   .BYTE 1
DATA:  .BYTE 256
```