```
;*********************************************
;RC 5 decoding programm
;device: ATtiny 13
;autor: Steven Weyrich
;*********************************************
.org 0x0000 ;adress for initial programm
    rjmp RESET
.org 0x0002 ;Adress for pin change interrupt
    rjmp PCINT0
.org 0x0006
    rjmp TIM0_COMPA ; Timer0 CompareA Handler

RESET:
ldi r16,$20 ;activate pull up resitor pin 5 (unused)
sts 0x36,r16
ldi r16,$02 ;initial values for TCCR0A
sts 0x4F,r16
ldi r16,$01 ;initial values for TCCR0B
sts 0x53,r16
ldi r16,$66 ;initial value Compare Register A
sts 0x56,r16
ldi r16,$20 ;Set Pin change enable in general interrupt mask register
sts 0x5B,r16
ldi r16,$10 ;Demask Pin change Interrupt for Pin 4
sts 0x35,r16
ldi r16,$04; demask output compaire match A interrupt
sts 0x59,r16
sts 0x37,r0 ;set all pins as imput tri-state
rjmp ini
;************************************************
;main programm
;************************************************

;***
;
;****

;****************************************************
; used registers:
;r0 : register for zero compaire
;r1 : imput register A
;r2 : imput register B
;r3 : first counter to enable power down mode
;r4 ; second sleep counting register
;r5 : RC 5 code start sequence test
;r16: copy register
;r17: recived 0 or 1 store register
;r18: floating pin blocking value
;r19: control word register
;r20: value for one full controll word
;r22: looplength
;r23: recived ones counting register
;r24: initial 1 test register
;r25: two bits counter
;r26: 04h
;r31: counter to enable sleep mode
;***************************************************

INI:
ldi r26,$04
clr r0      ;reset register for zero compaires
clr r3      ;reset first counting register
clr r4      ;reset second counting register
clr r5      ;reset RC 5 code start sequence test
clr r31     ;reset sleep counting register
clr r29     ;reset sleep counting register
ldi r18,$10 ;floating pin blocking value
;register auf null setzen muss noch einfefügt werden!
ldi r20,$01
sei         ;enable interrupt
rjmp MAIN

MAIN:
```

```
cp r5,r0        ;test sequence for initial 1 1
breq test       ;if startsbits not recived, idle sleep mode is activated, system waits
for new data or timer interrupt
;condition: startbits (1,1) recived, r5 non zero
cpi r25,$02
breq cw         ;controll word
cpi r20,$0C     ;if compaire possitive,the output routine is activated
brne rest
rjmp OUTPUT

test:
;this program part tests for the start bits 1,1
cpi r17,$0A ;start bits are coded as a change from high to low to high to low, r17 is s
hifted left after interrupt: 1010-->$A
brne rest
inc r5      ;r5 counts the received bits
clr r17
clr r6
clr r25
rjmp MAIN

;one and zero writes the received controll bits
cw:
cpse r17,r26
inc r19
lsl r19
clr r17
clr r6
clr r25
inc r20
rjmp MAIN

zero:
lsl r19
clr r17
inc r20
rjmp MAIN

rest:
ldi r16,$20     ;set sleep enable bit
sts 0x55,r16
sleep
ldi r16,$00     ;clear sleep enable bit
sts 0x55,r16
cpse r3,r0      ;first test for go to power down mode
rjmp MAIN
inc r4
cpse r4,r0      ;second test for go to power down mode
rjmp MAIN
clr r17         ;reset data register
ldi r16,$30     ;set power down mode and sleep enable bit
sts 0x55,r16
sleep
ldi r16,$00     ;clear sleep enable bit
sts 0x55,r16
rjmp INI

OUTPUT:
;OUTPUT activates the relais
andi r25,$81 ;deletes bits at position 1 and 8, because RC 5 only contains 6 controll b
its
clr r6       ;output time
cpi r19,$0C
breq out0
cpi r19,$0D
breq out1
cpi r19,$0F
breq out2
cpi r19,$10
breq out3
rjmp ini

out0:
```

```
sbi $16,0
ldi r16,$01
sts 0x37,r16
inc r6
rjmp ini

out1:
sbi $16,1
ldi r16,$02
sts 0x37,r16
inc r6
rjmp ini

out2:
sbi $16,2
ldi r16,$04
sts 0x37,r16
inc r6
rjmp ini

out3:
sbi $16,3
ldi r16,$08
sts 0x37,r16
inc r6
rjmp ini
;***************************************************************************
;Interrupt service routine                                                 *
;***************************************************************************

;***************************************************************************
;the idear is to detect the logic level change at the import pin. The      *
;imported PORTB is logical AND combined with 00100000 to prevent incorrect  *
;results                                                                   *
;When the signal leads to a pin change interrupt, the timer is in          *
;started in CTC mode. When OCR0A and the timer match, the new pin 4 value   *
;and the interrupt start value are compaired. When not equal, the controll  *
;register is deleted. Outerwise the counting register r17 is increased     *
;(when logical low for a RC 5 code higt) or not. After that the value is    *
;rolled to the left.                                                       *
;in case of 0 to 1 or the other way round, the timer creates to times to a  *
;CTC Interrupt                                                             *
;***************************************************************************

;***************************************************************************
;used registers:                                                           *
;r0 : register for zero compaire                                           *
;r1 : imput register A                                                     *
;r2 : imput register B                                                     *
;r16: copy register                                                        *
;r17: recived 0 or 1 store register                                        *
;r18: floating pin blocking value
;r20: first sleep counting register                                        *
;***************************************************************************

PCINT0:
IN r1,0x16  ;load Port B
and r1,r18  ;delete all non interessting Port B values
ldi r16,$20     ;set sleep enable bit
sts 0x55,r16
sts 0x52,r0 ;reset timer, nessesary for later interrupts
sei
sleep
sts 0x52,r0 ;reset timer, nessesary for later interrupts
reti

TIM0_COMPA:
IN r2,0x16  ;load Port B
and r2,r18  ;delete all non interessting Port B values
cpse r2,r1  ;compare if signal is stable (no noise)--> no change in signal
rjmp foult
cpi r20,$02 ;first sleep counting register, if more then two times signal has the same
value, its wrong
```

```
brsh foult
cpse r2,r18 ;high level test for the noninverted signal (low level at pin)!
inc r17     ;increased if original signal is high
lsl r17     ;safes correct value
inc r3
inc r25
sei
cpse r6,r0
rjmp so         ;stop output
sts 0x52,r0 ;reset timer
reti

foult:
clr r20
clr r17     ;clear received signal
sts 0x52,r0 ;reset clock
cpse r6,r0
rjmp so         ;stop output
sei
reti

so:
inc r7      ;counter for stop output
cpse r7,r0
reti
;sts 0x37,r0
dec r6
reti
```

```
.include "tn13def.inc"
.device ATtiny13

.def zero=r0
.def one=r1
.def sc1=r2
.def cr=r19
.def temp=r16
.def counter=r17
.def impulses=r18

.org 0
rjmp RESET

.org 3
rjmp TO

.org 6
rjmp CMA

.org 7
rjmp CMB

RESET:
inc one
lsl one
ldi temp,$40
out TCCR0A,temp
ldi temp,$02
out TCCR0B,temp
ldi temp,$03
out OCR0A,temp
ldi temp,$03
out TIMSK0,temp
ldi temp,$0F
out OCR0B,temp
sei

MAIN:
out PORTB,one   ;set Pin1 high
out TCNT0,zero  ;reset timer
rcall ONE1
rcall ONE1
rcall ONE1
ADDRESS:
cpi cr,$08
breq COMMAND
rcall ZERO1
rjmp ADDRESS
COMMAND:
rcall ZERO1
rcall ONE1
rcall ONE1
rcall ZERO1
rcall ONE1
rcall ZERO1
rcall WAIT
rjmp MAIN

TO:;timer overflow
;when the timer overflow interrupt occures sc1 (sleep counter 1) is increased.
;when sc1 has run throw its length (Time needed:255(register steps)*255(timer steps)*8(
Timer prescaler)/4.8MHz= 108,375 ms) sc2 is increesed
;time for one compled circle: 255(register steps)*255(register steps)*255(timer steps)*
8(timer prescaler)/4.8MHz=27,64 s
inc sc1
reti

CMA:;first timer compaire match interrupt
;the initial sending part set Pin 1 to high, the overflow sets it to low
;time needed from reseted timer to set Pin 1 low:
;[2(compaire value) * 8(timer prescaler)+4 cycles interupt response time+4cycles wake u
p from sleep+2 cycles rjmp+1 cycles program+5 cycles nop+ 1 cycle delay set Pin high an
d reset timer]/4,8MHz=6,875µs (69,4ns to fast)
nop
nop
nop
nop
nop
out PORTB,zero
reti

CMB:;second timer compaire match interrupt
;after a break when Pin 1 is set to low, the interrupt sets back to high
;time needed from CMA to CMB:
;[15(counter compaire value)*8(timer prescaler)+1 nop+2 instructions+4 interrupt respon
se+4 wake up+2 rjmp]/4.8MHz-CMA = 20,789µs (43,2ns to fast)
inc impulses
nop
out PORTB,one
out TCNT0,zero;reset timer
reti

ONE1:
;program creates a "1" (32 pulses low follow by 32 high). The symetric character of the
 rc5 code allowes to count
;pulses. the low pulses are created by the use of the data direction register as imputs
. The pin cange has no effect
;when Pin 1 is set as imput. for the following high pulses the direction register is se
t as output.
;ldi temp,$0C    ;demask Output compaire register A and B
;out TIMSK0,temp
out DDRB,zero   ;pin 1 as imput --> no signal is send
ONE2:
cpi impulses,$40 ;counter for the full word with 64 pulses length
breq END2
cpi impulses,$20 ;couter for the 32 pulses (32d =20h)
breq END1
ldi temp,$20
out MCUCR,temp
sleep
out MCUCR,zero
rjmp ONE2
END1:
out DDRB,one
rjmp ONE1
END2:
;ldi temp,$0A
;out TIMSK0,temp
clr impulses
inc cr
ret

ZERO1:
;ldi temp,$0C   ;demask Output compaire register A and B
;out TIMSK0,temp
out DDRB,one    ;pin 1 as output -->pulses are visible
zero2:
cpi impulses,$40
breq END4
cpi impulses,$20
breq END3
ldi temp,$20
out MCUCR,temp
sleep
out MCUCR,zero
rjmp ZERO2
END3:
out DDRB,zero
rjmp ZERO1
END4:
;ldi temp,$0A
;out TIMSK0,temp
clr impulses
inc cr
ret
```

```
WAIT:
;Timer prescaler: 1024*0,208µs*255=7311,36µs (x15)
;7311,36*15=109670,4µs -->Difference: 4107,6µs
;6*255*8*0,298= 3647,52µs--> difference: 460,08µs
;8x255x0,208=424,32µs difference: 35,76
;needed: 113,778 ms
ldi temp,$02
out TIMSK0,temp
ldi temp,$05
out TCCR0B,temp
clr sc1
out PORTB,zero
out TCNT0,zero
WAIT1:
ldi temp,$20
out MCUCR,temp
sleep
out MCUCR,zero
cpi sc1,$0F
brne WAIT1
ldi temp,$02
out TCCR0B,temp
clr sc1
out TCNT0,zero
WAIT2:
ldi temp,$20
out MCUCR,temp
sleep
out MCUCR,zero
cpi sc1,$06
brne WAIT2
ldi temp,$02
out TCCR0B,temp
clr sc1
out TCNT0,zero
WAIT3:
ldi temp,$20
out MCUCR,temp
sleep
out MCUCR,zero
cpi sc1,$08
brne WAIT3
ldi temp,$02
out TCCR0B,temp
out TCNT0,zero
ldi temp,$AA
out OCR0A,temp

ret
```