Contents

1 Opgave 1

1.1 Voorbereiding

1.1.1 Op welke pinnen zijn de 4 LED's aangesloten?

LED1 = PB0

LED2 = PB1

LED3 = PB2

VAUX-ENA = PC5

1.1.2 Hoe zijn ze aangesloten? Hoe moet ik ze aansturen?

Om een LED aan te zetten, moet je een 0 op de pin sturen en VAUX-ENA moet een 1 op zijn pin gekregen hebben.

1.1.3 GPIO pinnen schakelen als output

Dit vind je in de ATx Manual bij 29. Peripheral Module Address Map
Port A = 0x0600
Port B = 0x0620
Port C = 0x0640
Dit vind je bij 12.12 Register Description - Ports
Pinnen zetten als output:
DIR = +0x00 => Port A = 0x0600, Port B = 0x0620, Port C = 0x0640
Signaal aan geven
OUT = +0x04 => Port A = 0x0604, Port B = 0x0624, Port C = 0x0644

1.1.4 Code

```
;
; Opdracht1Ext.asm
;
; Created: 15/04/2022 15:07:49
; Author : Yorben
;
.CSEG
Init:
	ldi	r16,0b00100000; We zetten 5de bit op 1 voor verder VAUX-Enable op 1 te
zetten en PC5 als output te zetten
	sts	PORTC_DIR,r16; PC5 zetten als output
	sts	PORTC_OUT,r16; Een 1 zetten op VAUX-Enable
	ldi	r16,0b10000000; We zetten 7de bit op 1 voor PA7 als output te zetten
	sts	PORTA_DIR,r16; PA7 zetten als output
	ldi	r16,0b00000111; 0de - 2de bit op 1 voor PB0-2 als output te zetten
	sts	PORTB_DIR,r16; PB0-2 zetten als output
	ldi	r16,0b11111111; Elke led uitzetten
	sts	PORTA_OUT,r16; Elke led uitzetten
	sts	PORTB_OUT,r16; Elke led uitzetten
	ldi	r18,0; Delay
	ldi	r19,0; Delay
	ldi	r20,0; Delay

Main:
	ldi	r17,0; Deze gebruik ik voor de loop

Loop:
	ldi	r16,0b11111110; PB0 op 0 om eerste led aan te zetten
	sts	PORTB_OUT,r16; Waarde doorgeven aan de output
```

```
        call Delay

        ldi     r16,0b11111111; Alle leds uitzetten
        sts     PORTB_OUT,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b11111101; PB1 op 0 om tweede led aan te zetten
        sts     PORTB_OUT,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b11111111; Alle leds uitzetten
        sts     PORTB_OUT,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b11111011; PB2 op 0 om derde led aan te zetten en vorige uit
        sts     PORTB_OUT,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b11111111; Alle leds uizetten
        sts     PORTB_OUT,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b01111111; PA7 op 0 zetten om vierde led aan te zetten
        sts     0x604,r16; Waarde doorgeven aan de output

        call Delay

        ldi     r16,0b11111111; PA7 op 1 om vierde led uit te zetten
        sts     0x0604,r16; Waarde doorgeven aan de output

        call Delay

        inc     r17; We gaan continu r17 incrementen tot hij overflowt en de zero carry
flag wordt gezet, dan zetten we r17 terug op nul en beginnen opnieuw
        brne    Loop
        jmp     Main

Delay:
        inc     r18; We gaan continu r18 incrementen en branchen naar Delay tot r18
overflowt en de zero carry flag wordt gezet, dan wordt r19 terug op 0 gezet en gaan we
over brne Delay
        brne    Delay
        inc     r19; We gaan continu r19 incrementen en branchen naar Delay tot r19
overflowt en de zero carry flag wordt gezet, dan wordt r19 terug op 0 gezet en gaan we
over brne Delay
        brne    Delay
        inc     r20
        cpi     r20,5
        brne    Delay; We gaan continu r20 incrementen tot hij gelijk is aan 5, zoniet
branchen we terug naar Delay
        ldi     r18,0
        ldi     r19,0
        ldi     r20,0
        ret; Als we tot hier geraakt zijn springen we terug naar waar de delay is
opgeroepen
```

1.1.5 Uitleg Delay

De klok runt op 4MHz, dus er zijn 4.000.000 cycli nodig om een delay van 1 Hz te krijgen.

Instructies: 1. inc: 1 cycle

2. cpi: 1 cycle

3. brne: 1 if condition false, 2 if condition true

4. ldi: 1 cycle

5. ret: 4 cycles

$$4000000 = 512 * 256 * x + 512 * x + 3x^2 + 7$$

$$=> 3x^2 + 131584 * x - 3999993 = 0$$

Wat ruwweg uitkomt op een nulpunt van 30.

Dit komt neer op ongeveer 4.000.000/3.950.227= 1,0126Hz.

Als ik de leds time komt dit niet uit op 1Hz, dus mijn berekening is fout of mijn idee van de kloksnelheid klopt niet.

2 Opgave 2

2.1 Joystick

Up = PB3
Down = PB5
Center: PB7
Left: PB4
Right: PB6
Pull up weerstanden aanzetten: ob00011000 zie pagina 133

2.2 Code

```
;
; Opdracht2.asm
;
; Created: 15/04/2022 14:23:09
; Author : Yorben
;
Init:
; LED's
	ldi	r16,0b00100000; We zetten 5de bit op 1 voor verder VAUX-Enable op 1 te
zetten en PC5 als output te zetten
	sts	PORTC_DIR,r16; PC5 zetten als output
	sts	PORTC_OUT,r16; Een 1 zetten op VAUX-Enable
	ldi	r16,0b10000000; We zetten 7de bit op 1 voor PA7 als output te zetten
	sts	PORTA_DIR,r16; PA7 zetten als output
	ldi	r16,0b00000111; 0de - 2de bit op 1 voor PB0-2 als output te zetten
	sts	PORTB_DIR,r16; PB0-2 zetten als output
	ldi	r16,0b11111111; Elke led uitzetten
	sts	PORTA_OUT,r16; Elke led uitzetten
	sts	PORTB_OUT,r16; Elke led uitzetten
	; Joystick
	ldi	r16,0b00011000; Pull up weerstand aanzetten
	sts	PORTB_PIN3CTRL,r16; Pull up weerstand Up
	sts	PORTB_PIN4CTRL,r16; Pull up weerstand Left
	sts	PORTB_PIN5CTRL,r16; Pull up weerstand Down
	sts	PORTB_PIN6CTRL,r16; Pull up weerstand Right
	sts	PORTB_PIN7CTRL,r16; Pull up weerstand Center
	; Variabelen
	ldi	r18,0; Delay
	ldi	r19,0; Delay
	ldi	r20,0; Delay
	ldi	r27,5; DelayVariabele
	ldi	r21,1; Stapgrootte veranderen Delay
	ldi	r22,0; LedTeller
	ldi	r23,0b11111110; Initiele ledwaarden, we beginnen met richting links
	mov	r16,r23
	ldi	r24,1; Richting, we beginnen met richting links (1 = links, 2 = rechts)
	ldi	r25,0; Joystickwaarde

; Loopen door te zien op welke LED we zitten
Loop:
	cpi	r22,4; Zien of we op de laatste LED geraakt zijn
	brne	Verder; Zo niet gaan we gewoon verder
	ldi	r22,0; Zo ja dan resetten we de LedTeller
	mov	r16,r23; We starten terug bij de initiele ledwaarden

Verder:
	call	Joystick; Richting joystick uitlezen
	call	Output; Waardes doorgeven aan de LED's
	inc	r22; Teller van de Leds verhogen
```

```
        cpi     r24,1; We kijken of de richting gelijk is aan links
        breq    Links; Zo ja branchen we naar Links, anders gaan we verder
        cpi     r24,2; We kijken of de richting gelijk is aan rechts
        breq    Rechts; Zo ja gaan we branchen naar Rechts, anders gaan we verder
        jmp     Loop

Links:
        sec; Carry op 1 zetten anders gaan er twee leds aan, want de carry flag wordt
op bit 0 ingevoegd bij rol
        rol     r16; De carry flag wordt op bit 0 ingevoegd, waardoor alle bits 1 plek
naar links opschuiven
        call    Delay; We roepen de delay op
        jmp     Loop

Rechts:
        sec; Carry op 1 zetten anders gaan er twee leds aan, want de carry flag wordt
op bit 7 ingevoegd bij ror
        ror     r16; De carry flag wordt op bit 7 ingevoegd, waardoor alle bits 1 plek
naar rechts opschuiven
        call    Delay; We roepen de delay op
        jmp     Loop

Delay:
        inc     r18; We gaan continu r18 incrementen en branchen naar Delay tot r18
overflowt en de zero carry flag wordt gezet, dan wordt r19 terug op 0 gezet en gaan we
over brne Delay
        brne    Delay
        inc     r19; We gaan continu r19 incrementen en branchen naar Delay tot r19
overflowt en de zero carry flag wordt gezet, dan wordt r19 terug op 0 gezet en gaan we
over brne Delay
        brne    Delay
        inc     r20
        cp      r20,r27
        brne    Delay; We gaan continu r20 incrementen tot hij gelijk is aan r27, zoniet
branchen we terug naar Delay
        ldi     r18,0
        ldi     r19,0
        ldi     r20,0
        ret; Als we tot hier geraakt zijn springen we terug naar waar de delay is
opgeroepen

Output:
        sts     PORTB_OUT,r16;
        bst     r16,3; De 3de bit van r16 op T zetten want deze duid LED4 aan, maar om
die aan te sturen moeten we de waarde op bit 7 zetten
        bld     r26,7; De waarde die we opgeslagen hebben in T op de 7de bit zetten om
LED4 aan te sturen
        sts     PORTA_OUT,r26
        ret

; Joystick inlezen
Joystick:
        lds     r25,PORTB_IN; We lezen de Joystick waarde in
        ori     r25,0b00000111; De eerste 3 bits zijn van de LED's, de andere van de
Joystick. Dus willen we met die van de LED's geen rekening houden
        cpi     r25,0b11101111; We kijken of de 4de bit/PB4 op 0 staat, dit zou Links
aanduiden
        breq    RichtingLinks; Als dit inderdaad zo is branchen we naar RichtingLinks,
anders gaan we verder
        cpi     r25,0b10111111; We kijken of de 6de bit/PB6 op 0 staat, dit zou Rechts
aanduiden
```

```
	breq	RichtingRechts; Als dit inderdaad zo is branchen we naar RichtingRechts,
anders gaan we verder
	cpi	r25,0b11110111; We kijken of de 3de bit/PB3 op 0 staat, dit zou Boven
aanduiden
	breq	RichtingBoven; Als dit inderdaad zo is branchen we naar RichtingBoven,
anders gaan we verder
	cpi	r25,0b11011111; We kijken of de 5de bit/PB5 op 0 staat, dit zou Onder
aanduiden
	breq	RichtingOnder; Als dit inderdaad zo is branchen we naar RichtingOnder,
anders gaan we verder
	cpi	r25,0b01111111; We kijken of de 7de bit/PB7 op 0 staat, dit zou Center
aanduiden
	breq	RichtingCenter; Als dit inderdaad zo is branchen we naar RichtingCenter,
anders gaan we verder
	ret; We gaan terug naar waar we Joystick hebben opgeroepen

; Richting veranderen naar links
RichtingLinks:
	ldi	r23,0b11111110; Initiele ledwaarden aanpassen rekening houdend met de
richting
	ldi	r24,1; De richting zetten op links
	jmp	Joystick; Teruggaan naar Joystick

; Richting veranderen naar rechts
RichtingRechts:
	ldi	r23,0b11110111; Initiele ledwaarden aanpassen rekening houdend met de
richting, hier nemen we bit 3 ipv 7, door de logica in Output
	ldi	r24,2; De richting zetten op rechts
	jmp	Joystick; Teruggaan naar Joystick

; Delay verlagen/Snelheid verhogen
RichtingBoven:
	sub	r27,r21; De stapgrootte (r21) van de delayvariabele (r20) aftrekken
	jmp	Joystick; Teruggaan naar Joystick

; Delay verhogen/Snelheid verlagen
RichtingOnder:
	add	r27,r21; De delayvariabele (r20) optellen met de stapgrootte (r21)
	jmp	Joystick; Teruggaan naar Joystick

; Hier zetten we alles terug op de basis waarden
RichtingCenter:
	ldi	r24,1; Terug op richting links zetten
	ldi	r27,5; DelayVariabele terug resetten
jmp	Joystick
```

3 Opgave 3

RGB is verbonden aan PL9823-CS = PA6, USARTD-MOSI = PD3, VAUX-ENA = PC5

Klok staat op 32MHz = 0,03125us per cycle

3.1 Timing voor een 0

T0H = 0,35us => 0,35/0,03125 = 11,2 dus 11 cycli => 9 nop want 1 ldi en 1 sts

T0L = 1,36us => 1,36/0,03125 = 43,52 dus 43 cycli => 38 nop want 1 inc en 1 ret

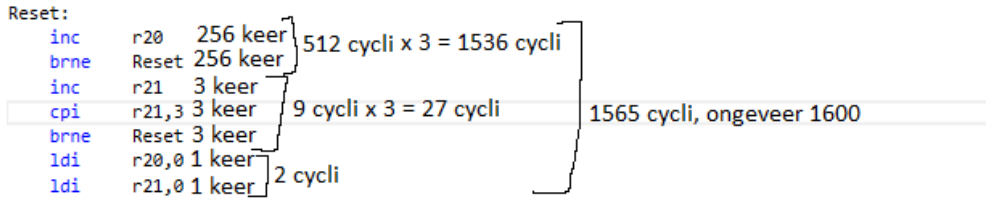
3.2 Timing voor een 1

T1H = 1,36us => 1,36/0,03125 = 43,52 dus 43 cycli => 38 nop want 1 inc en 1 ret

T1L = 0,35us => 0,35/0,03125 = 11,2 dus 11 cycli => 9 nop want 1 ldi en 1 sts

3.3 Timing voor Reset

RES = 50us => 50/0,03125 = 1600 dus 1600 cycli



$$1600 = 512x + 3x^2 + 2$$

$$=> 3x^2 + 512x - 1598 = 0$$

Komt ruwweg uit op een nulpunt van 3.

3.4 Kleur

Ik wil groen, waardoor ik dus eerst 8 nullen, dan 8 keer 1 en dan terug 8 nullen moet sturen.

3.5 Code

```
;
; Opdracht3.asm
;
; Created: 01/05/2022 19:51:21
; Author : yorbe
;
;***********************************************************************
       ; init oscillator:   external xtal(16MHz) en pll*2  => 32Mhz clock

       ; OSC.XOSCCTRL=0b11001011                => externe 16MHz clock
       ldi           r16, 0b11001011
       sts           osc_xoscctrl, r16
       // enable external oscillator
       ldi           r16,0b01000
       sts           osc_ctrl,r16
test1:
//check status off xoscrdy (bit3) ....wait until xos is ready
       lds           r16,osc_status
       sbrs   r16, 3        ; skip next instr if bit is set
       rjmp   test1
       // select xosc as source for PLL
       ldi           r16, 0b11000010
       sts           osc_pllctrl, r16
       // enable PLL and external oscillator
       ldi           r16,0b00011000
       sts           osc_ctrl,r16
test2:
//check status off PLL (bit4)    ... wait until PLL is ok
       lds           r16,osc_status
       sbrs   r16, 4        ; skip next instr if bit is set
       rjmp   test2
       //enable ccp
       ldi           r16, 0xd8
       sts           cpu_ccp,r16
       ldi           r16, 4
       sts           CLK_CTRL,r16         ; select PLL as clock (32MHz)
;***********************************************************************
```

```
.CSEG

Init:
	; USARTD-MOSI
	ldi	r16,0b00001000; 3de bit op 1 om PC3/USARTD-MOSI als output te zetten
	sts	PORTD_DIR,r16; P3 zetten als output
	; PL9823-CS
	ldi	r16,0b01000000; 6de bit op 1 om PA6/PL9823-CS als output te zetten
	sts	PORTA_DIR,r16; PA6 als output zetten
	sts	PORTA_OUT,r16; Een 1 zetten op PL9823-CS, om de MOSFET te laten geleiden
	; VAUX-ENA
	ldi	r16,0b00100000; 5de bit op 1 om PC5/VAUX-ENA weer te geven
	sts	PORTC_DIR,r16; PC5 als output zetten
	sts		PORTC_OUT,r16; Een 1 zetten op VAUX-ENA



	; Variabelen
	ldi		r17,0; BitTeller
	ldi		r18,0; LedTeller
	ldi		r20,0; ResetTeller
	ldi		r21,0; ResetTeller

; Reset geven van 50us
Reset:
	inc		r20; We gaan continu r20 incrementen en branchen naar Reset tot
r20 overflowt en de zero carry flag wordt gezet, dan wordt r20 terug op 0 gezet en
gaan we over brne Reset
	brne	Reset
	inc		r21
	cpi		r21,3; We gaan continu r21 incrementen tot hij gelijk is aan 3,
zoniet branchen we terug naar Reset
	brne	Reset
	ldi		r20,0
	ldi		r21,0

Rood:
	call	Nul; Ik wil groen, dus deze bitstream moet een 0 geven
	cpi		r17,8
	brne	Rood; Zolang we niet 8 keer erdoor zijn gegaan blijven we branchen naar
Rood
	ldi		r17,0; We resetten de teller

Groen:
	call	Een; Ik wil groen, dus deze bitstream moet een 1 geven
	cpi		r17,8; We moeten dit 8 keer doen
	brne	Groen; Zolang we niet 8 keer erdoor zijn gegaan blijven we branchen naar
Groen
	ldi		r17,0; We resetten de teller

Blauw:
	call	Nul; Ik wil groen, dus deze bitstream moet een 0 geven
	cpi		r17,8; We moeten dit 8 keer doen
	brne	Blauw; Zolang we niet 8 keer erdoor zijn gegaan blijven we branchen naar
Blauw
	ldi		r17,0; We resetten de teller

Leds:
	inc		r18
	cpi		r18,4; We moeten alle RGB Leds doorlopen hebben
	brne	Rood; Als dit niet het geval is doorlopen we de kleuren opnieuw voor de
volgende LED
```

```
Loop:
	jmp		Loop; Kleur wordt niet verandert dus we blijven gewoon doorloopen

Nul:
	ldi		r16,0b00001000; Signaal hoog zetten
	sts		PORTD_OUT,r16; Signaal doorgeven aan de MOSFET
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	ldi	r16,0b00000000; Signaal laag zetten
	sts	PORTD_OUT,r16; Signaal doorgeven aan de MOSFET
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	nop
	inc	r17
	ret

Een:
	ldi	r16,0b00001000; Signaal hoog zetten
	sts	PORTD_OUT,r16; Signaal doorgeven aan de MOSFET
```

```
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
ldi     r16,0b00000000; Signaal laag zetten
sts     PORTD_OUT,r16; Signaal doorgeven aan de MOSFET
nop
nop
nop
nop
nop
nop
nop
nop
nop
inc     r17
ret
```