Inhoud

DIRSET: Individuele pinnen als uitgang zetten

DIRCLR: Individuele pinnen als ingang zetten

1 Opgave 1

C declareert het einde van een string met \0. Doordat Counter%d\r (%d en \r nemen elk 1 plaats in) al 10 plaatsen inneemt wordt deze \0 naar de volgende lijn gebracht. Dus wordt deze op de plaats van a gezet en override deze, waardoor die continu terug op 0 gezet wordt. Dit kan dus opgelost worden door de grootte van variabele text aan te passen.

2 Opgave 2

Het niveau op de GPIO pinnen moet laag zijn om de LED's te doen branden, doordat ze aan de andere kant al een voeding hebben genaamd 3,3VAux.

3 Opgave 3

Om de output van de schakelaar te kunnen zien heb je een pull up weerstand nodig.

PB3 = Up

PB4 = Left

PB5 = Down

PB6 = Right

PB7 = Center

4 Opgave 4

4.1 Configureer H-brug lijnen als output

PORTF.DIRSET = 0b00011111: Eerste 3 bits voor CTRL pinnen (PF0-3), 4de bit voor Sleep pin (PF4) en 5de bit voor Fault pin (PF5)

4.2 Configureer TCF0 in single slope PWM

TCF0.CTRLA = 0b00000001: Laatste 4 bits zijn gereserveerd (zie pg. 154) en eerste 4 bits 0001 om prescaler te delen door 1, voor de hoogst mogelijke frequentie, anders draait motor niet bij lagere getallen.

TCF0.CTRLB = 0b11110011: Eerste 3 bits bepalen Waveform Generation Mode dus 011 voor single slope (pg.155), 4de bit is gereserveerd dus een 0 zetten (pg. 154) en laatste 4 bits moeten op 1 staan om CC kanalen in te schakelen (pg.154 en dia 58).

TCF0.CTRLD = 0b00000000: Eerste 3 bits 0 want Event Action moet uitstaan (dia 58 en pg. 156), de andere bits zetten we ook gewoon uit.

TCF0.PER = 4095: Periode (zie parameters DriverMotorSet)

4.3 Schrijf de functie DriverMotorSet()

Stijgende flanken worden bepaald door PER en dalende flanken door CCx.

Hogere CCx waarde betekent kleinere dutycycle, waardoor de golf minder vaak van waarde verandert en dus het gemiddelde voltage hoger ligt en dus de snelheid ook hoger is.

CCA

CCB

CCC

CCD

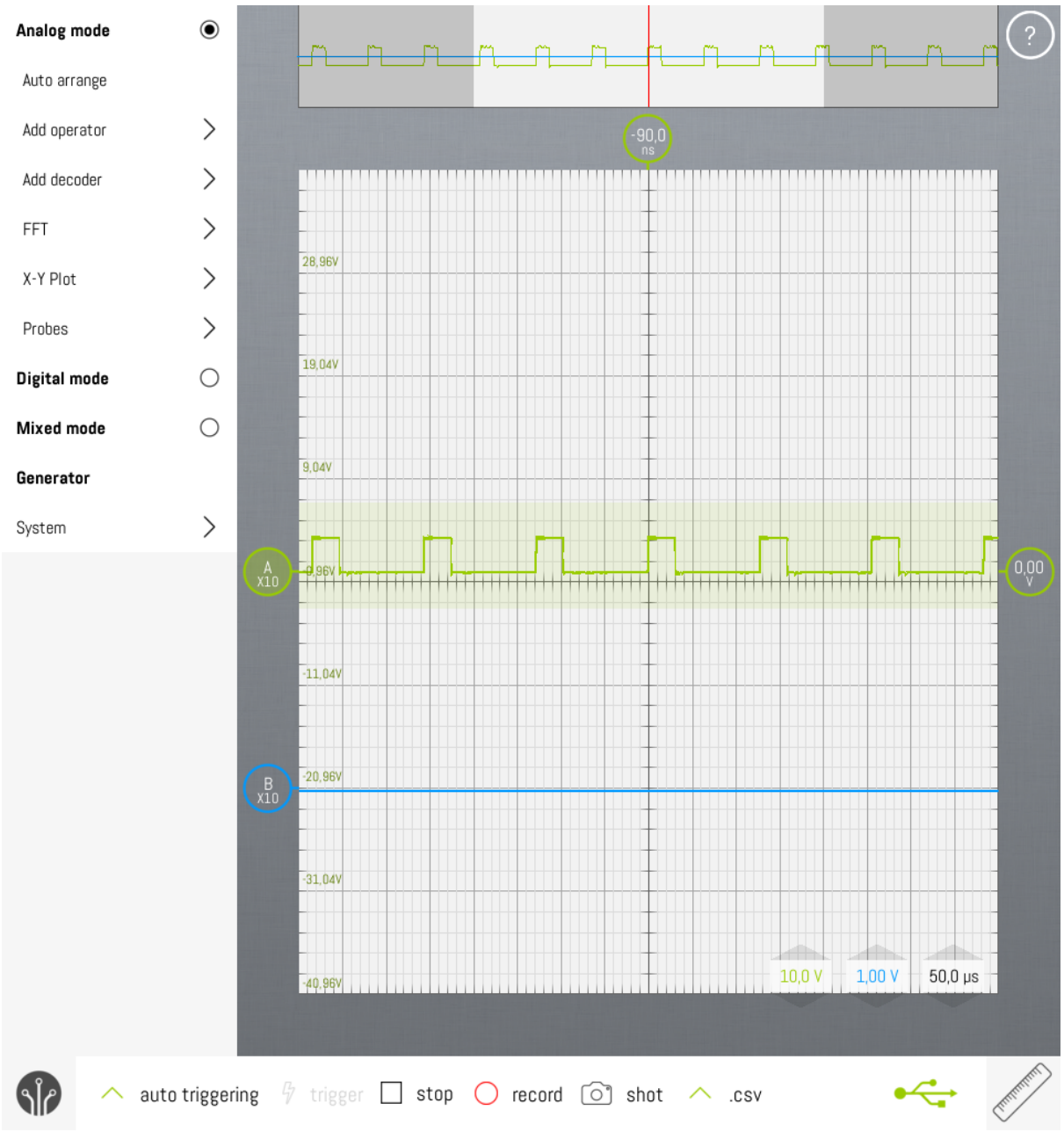
4.4 Oscilloscoop

T22 = MOTOR2-CTRL1 = ENC2A

T23 = MOTOR1-CTRL1 = ENC1A

T24 = MOTOR2-CTRL2 = ENC2B

T30 = MOTOR1-CTRL2 = ENC1B



5 Opgave 5

5.1 Enable interrupts dia 67

SREG = 0b10000000: Zet Global Interrupt Enable (GIE) aan (pg.17).

PMIC.CTRL = 0b00000111: Bit 0 is om Low-level Interrupts aan te zetten, bit 1 voor Medium-level Interrupts, bit 2 voor High-level Interrupts, bit 3-5 zijn gereserveerd dus 0 en van bit 6-7 blijven we af (pg.118)

5.2 Configureer de 4 encoder lijnen

5.2.1 Encoder 1

5.2.1.1 Initialisatie

PC6 en PC7

PORTC.DIR = 0b11000000: PC6 en PC7 als uitgang zetten

PORTC.INTCTRL = 0b00001111: Eerste 4 bits tonen het interrupt level aan (11 staat voor High-level interrupt pg.115), laatste 4 bits zijn gereserveerd dus moeten op 0 (pg.131)

PORTC.INT0MASK = 0b01000000: Duid aan welke poort er voor deze interrupt gebruikt wordt, hier de 6de bit dus PC6

PORTC.INT1MASK = 0b10000000: Duid aan welke poort er voor deze interrupt gebruikt wordt, hier de 7de bit dus PC7

PORTC.PIN6CTRL = 0b00000000: Eerste 3 bits duiden aan op welke edge de interrupt sensed (000, pg.134), de volgende 3 bits duiden de output/pull configuration aan hier dus gewoon TOTEM (000, pg.133), bit 6 is voor te inverteren, maar dit willen we niet dus nemen we o (pg.133) en de laatste bit is gereserveerd dus ook 0 (pg.133).

PORTC.PIN7CTRL = 0b00000000: Eerste 3 bits duiden aan op welke edge de interrupt sensed (000, pg.134), de volgende 3 bits duiden de output/pull configuration aan hier dus gewoon TOTEM (000, pg.133), bit 6 is voor te inverteren, maar dit willen we niet dus nemen we o (pg.133) en de laatste bit is gereserveerd dus ook 0 (pg.133).

5.2.1.2 ISR

Vector naam = PORTC.INT0_vect en PORTC.INT1_vect (dia 70).

5.2.2 Encoder 2

5.2.2.1 Initialisatie

PE4 en PE5

PORTE.DIR = 0b00110000: PE4 en PE5 als uitgang zetten

PORTE.INTCTRL = 0b00001111: Eerste 4 bits tonen het interrupt level aan (11 staat voor High-level interrupt pg.115), laatste 4 bits zijn gereserveerd dus moeten op 0 (pg.131)

PORTE.INT0MASK = 0b00010000: Duid aan welke poort er voor deze interrupt gebruikt wordt, hier de 4de bit dus PC4

PORTE.INT1MASK = 0b00100000: Duid aan welke poort er voor deze interrupt gebruikt wordt, hier de 5de bit dus PC5

PORTE.PIN4CTRL = 0b00000000: Eerste 3 bits duiden aan op welke edge de interrupt sensed (000, pg.134), de volgende 3 bits duiden de output/pull configuration aan hier dus gewoon TOTEM (000, pg.133), bit 6 is voor te inverteren, maar dit willen we niet dus nemen we o (pg.133) en de laatste bit is gereserveerd dus ook 0 (pg.133).

PORTE.PIN5CTRL = 0b00000000: Eerste 3 bits duiden aan op welke edge de interrupt sensed (000, pg.134), de volgende 3 bits duiden de output/pull configuration aan hier dus gewoon TOTEM (000, pg.133), bit 6 is voor te inverteren, maar dit willen we niet dus nemen we o (pg.133) en de laatste bit is gereserveerd dus ook 0 (pg.133).

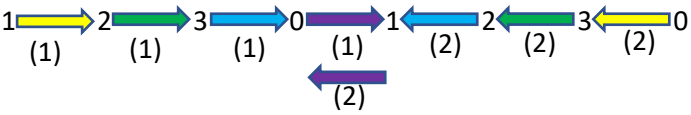
5.2.2.2 ISR

Vector naam = PORTE.INT0_vect en PORTE.INT1_vect (dia 70).

5.3 Encoder uitleg

Spoor 1:	1	1	0	0	1	1	0	0 …
Spoor 2:	0	1	1	0	0	1	1	0 …

Gray code:



Optellen:	1 -> 2 (1):	1	1 (4b)	**Aftrekken:**	0 -> 3 (2):	0	0 (4a)	
		0	1			0	1	
	2 -> 3 (1):	1	0 (1b)		3 -> 2 (2):	0	1 (2b)	
		1	1			1	1	
	3 -> 0 (1):	0	0 (3a)		2 -> 1 (2):	1	1 (3b)	
		1	0			1	0	
	0 -> 1 (1):	0	1 (2a)		1 -> 0 (2):	1	0 (1a)	
		0	0			0	0	

Interrupt 1 (Spoor 1 verandert):

1. Spoor 1 verandert naar 0 (1)
    a. Spoor 2 is 0: Optellen (a)
    b. Spoor 2 is 1: Aftrekken (b)
2. Spoor 1 verandert naar 1 (2)
    a. Spoor 2 is 0: Optellen (a)
    b. Spoor 2 is 1: Aftrekken (b)

Interrupt 2 (Spoor 2 verandert):

1. Spoor 2 verandert naar 0 (3)
    a. Spoor 1 is 0: Optellen (a)
    b. Spoor 1 is 1: Aftrekken (b)
2. Spoor 2 verandert naar 1 (4)
    a. Spoor 1 is 0: Aftrekken (a)
    b. Spoor 1 is 1: Optellen (b)

Resulterende code:

```
ISR(PORTE_INT0_vect) { // Deze wordt opgeroepen als PE4/ENC2A verandert
	if ((PORTE.IN & 0b00010000) == 0) { // Door de AND functie kunnen we zien of
bit4/ENC2A/PE4 op 0 staat
		if ((PORTE.IN & 0b00100000) == 0) { // Door de AND functie kunnen we
zien of bit4/ENC2B/PE5 op 0 staat
			// Als ENC2A op 0 staat en ENC2B op 0 moeten we aftrekken
			Encoder1--;
		}
		// Als ENC2A op 0 staat en ENC2B op 1 moeten we optellen
		else {
			Encoder1++;
		}
	}
	else {
		if ((PORTE.IN & 0b0010000) == 0) { // Door de AND functie kunnen we zien
of bit5/ENC2B/PE5 op 0 staat
			// Als ENC2A op 1 staat en ENC2B op 0 moeten we optellen
			Encoder1++;
		}
		// Als ENC2A op 1 staat en ENC2B op 1 moeten we aftrekken
		else {
			Encoder1--;
		}
	}
}

ISR(PORTE_INT1_vect) { // Deze wordt opgeroepen als PE5/ENC2B verandert
	if ((PORTE.IN & 0b00100000) == 0) { // Door de AND functie kunnen we zien of
bit5/ENC2B/PE5 op 0 staat
		if ((PORTE.IN & 0b00010000) == 0) { // Door de AND functie kunnen we
zien of bit4/ENC2A/PE4 op 0 staat
			// Als ENC2A op 0 staat en ENC2B op 0 moeten we optellen
			Encoder1++;
		}
		// Als ENC2A op 0 staat en ENC2B op 1 moeten we aftrekken
		else {
			Encoder1--;
		}
	}
	else {
		if ((PORTE.IN & 0b00010000) == 0) { // Door de AND functie kunnen we
zien of bit4/ENC2B/PE4 op 0 staat
			// Als ENC2A op 1 staat en ENC2B op 0 moeten we aftrekken
			Encoder1--;
		}
		// Als ENC2A op 1 staat en ENC2B op 1 moeten we optellen
		else {
			Encoder1++;
		}
	}
}
```

6 Opgave 6

6.1 Baudrate instellen

Pg.262

BSEL = 2094 = 100000101110 (12 bits)

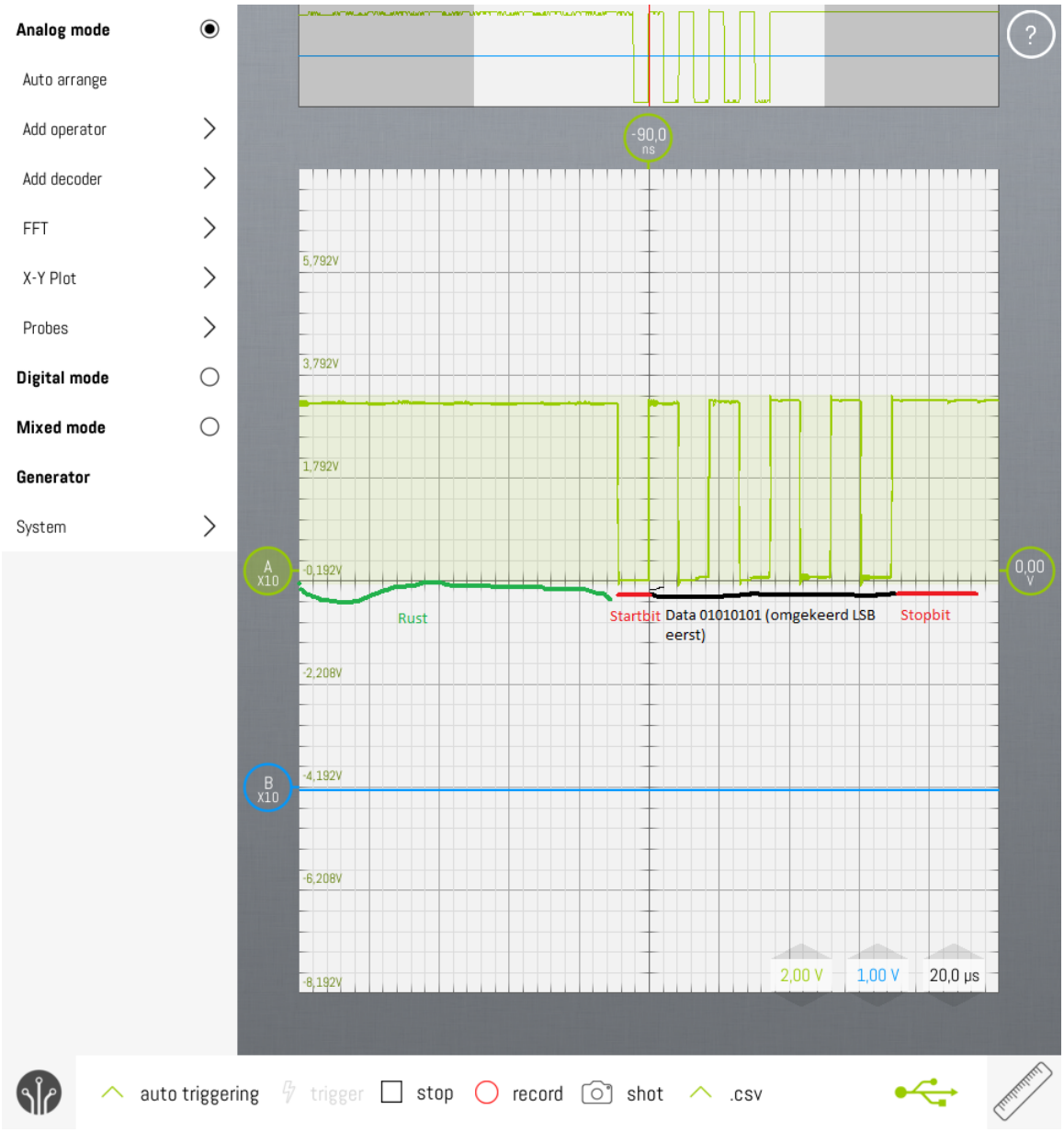
BSCALE = -7 = 1001 (4 bits)

USART.BAUDCTRLA = 0b00101110 = 0x2E: Eerste 8 bits van BSEL beginnend van de LSB

USART.BAUDCTRLB = 0b10011000 =0x98: Eerste 4 bits zijn de laatste 4 bits van BSEL en de laatste 4 bits zijn BSCALE

6.2 Zend 0x55 over

0x55 = 01010101



7 Opgave 7

7.1 DriverADCInit

pg.290 en 299

Opdracht: interne spanningsreferentie 1V, 12 bit left adjusted mode en geen freerun

DriverADCInit():

ADCA.CTRLA = 0b00000111: Bit 0 enabled ADC, bit 1 Flushed ADC, bit 2 start ADC conversie en laatste 4 bits zijn gereserveerd

ADCA.CTRLB = 0b00010110: Bit 0 is gereserveerd dus 0, bit 1 en 2 bepalen de resolutie (12 voor het grootste bereik ppt), bit 3 bepaalt free run mode, bit 4 de conversie modus (signed voor grootste bereik ppt), bit 5-6 00 voor geen limiet en bit 7 is gereserveerd

ADCA.REFCTRL = 0b00000010: Bit 0 op 0 want geen temperatuur sensor, bit 1 enables bandgap, bit 2-3 zijn gereserveerd dus 0, bit 4-6 bepalen de referentie dus 000 (bandgap) en bit 7 is gereserveerd dus 0

ADCA.PRESCALER = 0b00000011: Bit 0-2 bepaalt de clock van ADC mag max 1,8MHz zijn dus we delen door 32 om hier onder te zitten en toch een redelijk grote frequentie te hebben, de andere bits zijn gereserveerd dus 0

ADCA.CH0.INTCTRL = 0b00000000: Bit 0-1 bepaalt interrupt niveau dus 00 voor off, bit 2-3 de interrupt mode 00 voor complete en bit 4-7 zijn gereserveerd dus 0

DriverADCGetCh():

Niet afgekregen