
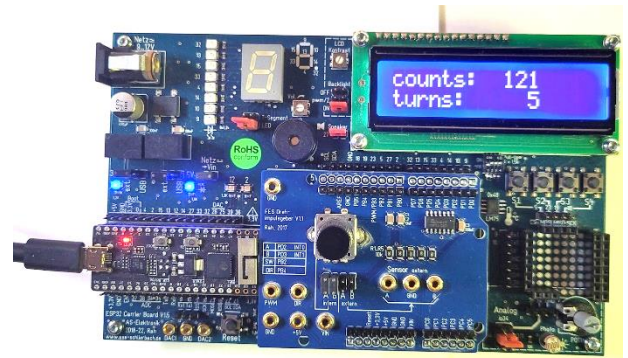


| | | |
|---|------------------------------------|--|
|  | IT: Hardwarenahes Programmieren | Name: Rahm Datum: 22.10.2022 1.7.2_Drehrichtungserkennung_mit_Encoder.docx |
| | Programmierung: Externer Interrupt | 1.7.2.1 |

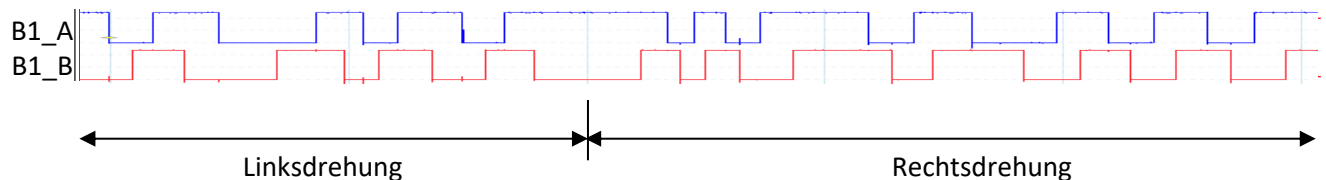
Projekt: Drehrichtungserkennung mit Interrupt

Ein Drehimpulsgeber (Drehencoder) gibt abhängig von der Drehung eine bestimmte Anzahl Impulse aus. Durch Zählen der Impulse kann die Drehung ermittelt werden.

Zur Unterscheidung der Drehrichtung werden zwei um 90° versetzte Rechtecksignale verwendet.



Impulsfolge bei Rechts-/Links Drehbewegungsumkehr



Arbeitsauftrag

- Das nebenstehende Programm zählt die Impulse auf Kanal A des Encoders in der Interruptroutine hoch. Analysieren Sie die Funktionsweise:

Wieviele Impulse (counts) können maximal gemessen werden?

Wie wird der Zählvorgang ausgelöst?

- Messen Sie die Gebersignale mit dem Oszilloskop nach.
- Mit Hilfe von Kanal B soll nun eine Drehrichtungserkennung realisiert werden. Ergänzen Sie die Interrupt-Routine so, dass der Zähler (counts) bei Rechtsdrehung hoch- und bei Linksdrehung runterzählt.
- Der Zähler soll nun mit dem Taster des Drehgebers zurückgesetzt werden können. (Zum Rücksetzen wird kein Interrupt benötigt.)
- Dokumentieren Sie Ihre Ergebnisse.

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

const int B1_A = 14;
const int B1_B = 4;

volatile uint16_t counts = 0;
uint16_t turns;

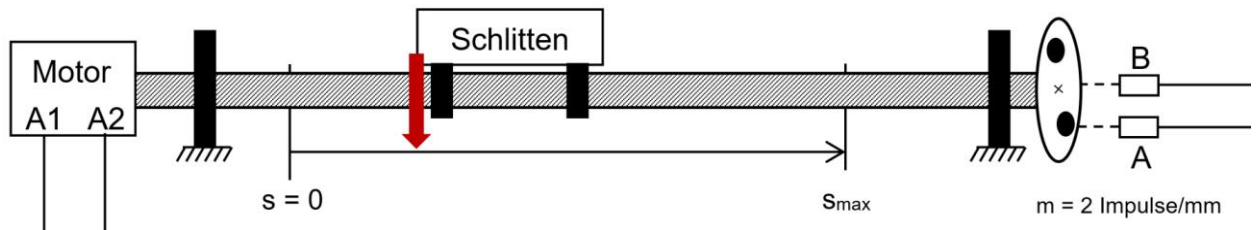
void IRAM_ATTR channelA_isr(void)
{
    counts++;
}

void setup()
{
    Wire.begin(21, 22);
    lcd.init();
    lcd.clear();
    lcd.print("counts: ");
    lcd.setCursor(0, 1);
    lcd.print("turns: ");

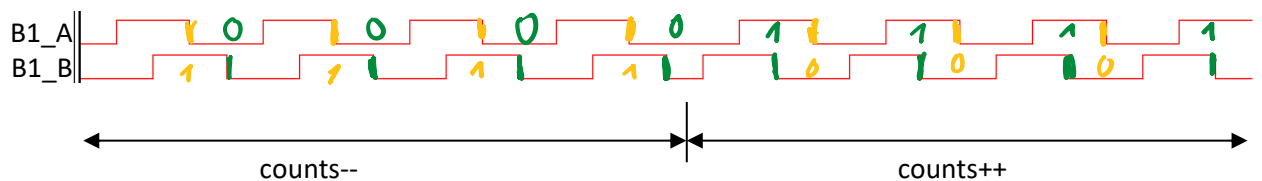
    pinMode(B1_A, INPUT_PULLUP);
    pinMode(B1_B, INPUT_PULLUP);
    attachInterrupt(B1_A, channelA_isr, FALLING);
}

char buf[10];
void loop()
{
    lcd.setCursor(7, 0);
    sprintf(buf, "%5d", counts);
    lcd.print(buf);
    turns = counts/24; //24 Impulse pro Umdrehung
    sprintf(buf, "%5d", turns);
    lcd.setCursor(7, 1);
    lcd.print(buf);
}
```

Projekt: Lageregelstrecke mit Inkrementalsensor

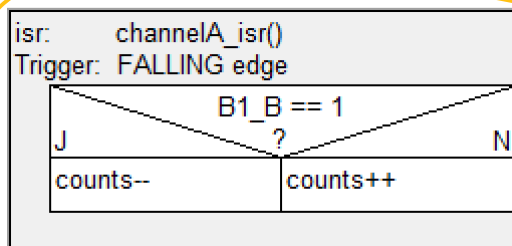


Impulsfolge bei Rechts-/Links Bewegungsumkehr



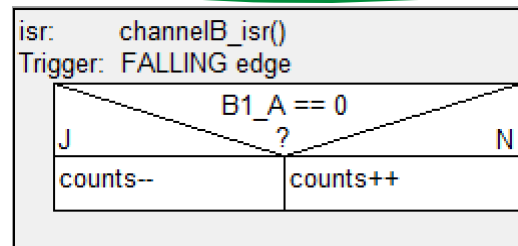
Ein inkrementaler Drehgeber funktioniert nach dem gleichen Prinzip wie der Drehencoder. Der Unterschied ist nur, dass er keine Raststellungen hat. Daher kann beim Drehgeber durch die zusätzliche Auswertung der negativen Flanke von Signal B die Auflösung verdoppelt werden. Der gezeigte Algorithmus verwendet dazu einen zweiten externen Interrupt für Signal B.

2 Quadranten Encoder



```
void IRAM_ATTR channelA_isr(void)
{ // Interrupt bei fallender Flanke an A
  if (digitalRead(B1_B) == HIGH)
    counts--;
  else
    counts++;
}
```

1Q



```
void IRAM_ATTR channelB_isr(void)
{ // Interrupt bei fallender Flanke an B
  if (digitalRead(B1_A) == LOW)
    counts--;
  else
    counts++;
}
```

2Q

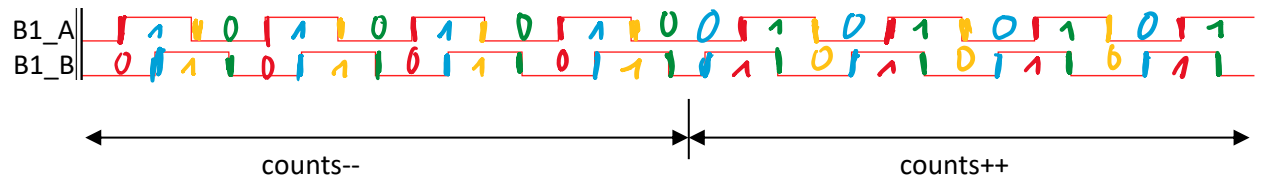
Arbeitsauftrag

Ändern Sie Ihr Programm für den Drehencoder so ab, dass pro Raststellung 2 counts gezählt werden. Alternativ schließen Sie die den Drehgeber der Lageregelstrecke an.

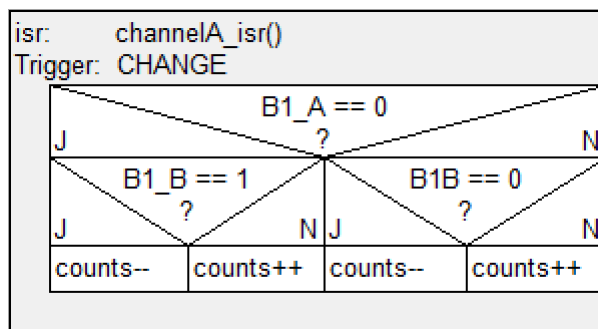
4 Quadranten Encoder

Die Auflösung kann nochmals verdoppelt werden, wenn nicht nur die negativen Flanken, sondern auch die positiven Flanken von A und B ausgewertet werden. Dafür muss die Zweiflanken-Triggerrung (CHANGE) für die beiden externen Interrupts aktiviert werden:

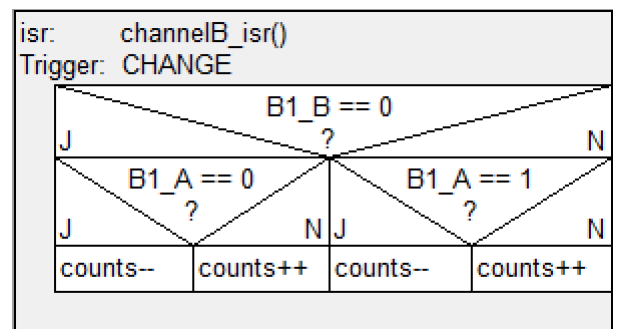
Impulsfolge bei Rechts-/Links Bewegungsumkehr



Struktogramm des Algorithmus



Handwritten annotations for channelA_isr: 1Q (yellow bracket under Dec1) and 3Q (red bracket under Inc1, Dec2, Inc2).



Handwritten annotations for channelB_isr: 2Q (green bracket under Dec3, Inc3) and 4Q (blue bracket under Dec4, Inc4).

```

    void IRAM_ATTR channelA_isr(void)
    {
      if (digitalRead(B1_A) == LOW)
      {
        // Interrupt bei fallender Flanke an A
        if (digitalRead(B1_B) == HIGH)
          counts--;
        else
          counts++;
      }
      else
      {
        //Interrupt bei steigender Flanke an A
        if (digitalRead(B1_B) == LOW)
          counts--;
        else
          counts++;
      }
    }
  
```

Handwritten annotation: 1Q (yellow circle around the first if block)

Handwritten annotation: 3Q (red circle around the second if block)

```

    void IRAM_ATTR channelB_isr(void)
    {
      if (digitalRead(B1_B) == LOW)
      {
        // Interrupt bei fallender Flanke an B
        if (digitalRead(B1_A) == LOW)
          counts--;
        else
          counts++;
      }
      else
      {
        //Interrupt bei steigender Flanke an B
        if (digitalRead(B1_A) == HIGH)
          counts--;
        else
          counts++;
      }
    }
  
```

Handwritten annotation: 2Q (green circle around the first if block)

Handwritten annotation: 4Q (blue circle around the second if block)

Arbeitsauftrag

Ändern Sie Ihr Programm für den Drehencoder so ab, dass pro Raststellung 4 counts gezählt werden.