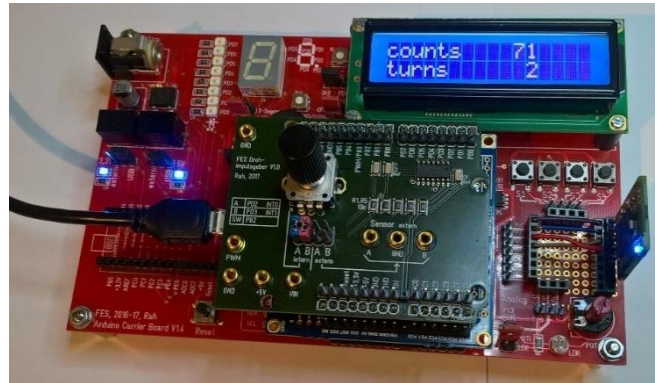
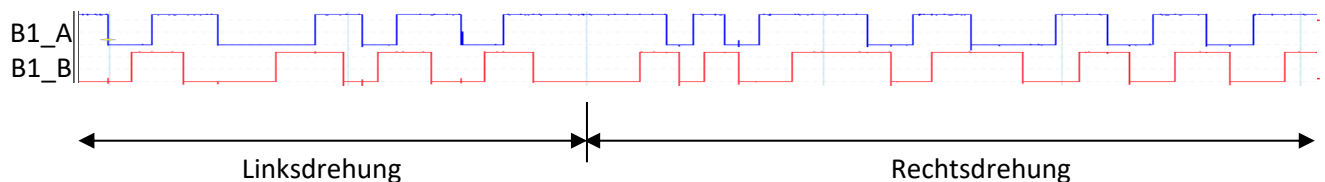
 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 17.12.2017 4_2_1_Inkrementaler_Drehgeber_mit_Externem_Interrupt.docx.docx
	Inkrementaler Drehimpulsgebers	4.2.1.1

Projekt: Drehrichtungserkennung mit Interrupt

Ein Drehimpulsgeber 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 (PD2) des Impulsgebers 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 (PD3) soll nun eine Drehrichtungserkennung realisiert werden. Ergänzen Sie die Interrupt-Routine so, dass der Zähler bei Rechtsdrehung hoch- und bei Linksdrehung runterzählt.
- Erstellen Sie ein Versuchsprotokoll und dokumentieren Sie Ihre Ergebnisse.



Für Profis: Der Zähler soll mit dem Taster des Drehgebers zurückgesetzt werden können.

```
#include "controller.h"

#define m 24 // Impulse/Umdrehung

volatile uint16_t counts; // Zähler


void setup (void)
{ /* Initialisierungen */
  ext_interrupt_init(ext_interrupt_isr);
  lcd_init();
  lcd_clear();
  lcd_setcursor(1,1);
  lcd_print("counts =");
  lcd_setcursor(2,1);
  lcd_print("turns =");
}

void main(void) // Funktion main()
{
  uint16_t turns;

  setup();
  ext_interrupt_enable(); // Interrupt freigeben

  while(1) // Endlosschleife loop()
  {
    lcd_setcursor(1,8);
    lcd_int(counts);
    turns = counts / m;
    lcd_setcursor(2,8);
    lcd_int(turns);
  }
}

void ext_interrupt_isr(void)
{
  counts++;
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller	Name: Rahm Datum: 17.12.2017 4_2_1_Inkrementaler_Drehgeber_mit_Externem_Interrupt.docx.docx
	Inkrementaler Drehimpulsgebers	4.2.1.2

Lösung:

```
#include "controller.h"

#define Sensor    _PORTD_
#define B1_A      2           // INT0
#define B1_B      3           // INT1
#define Taster    _PORTB_
#define B1_SW     2           // Taster des Drehgebers ist Low-aktiv
#define m         24          // Impulse/Umdrehung
#define MAXCOUNTS 2000

volatile uint16_t counts;      // Variablen in ISR immer als volatile !!

void setup (void)
{ /* Initialisierungen */
  bit_init(Sensor,B1_B,IN);    // Impulse/Umdrehung B1_B ohne Interrupt!
  bit_init(Taster,B1_SW,IN);

  lcd_init();
  lcd_clear();
  lcd_setcursor(1,1);
  lcd_print("counts =");
  lcd_setcursor(2,1);
  lcd_print("turns =");

  ext_interrupt_init(ext_interrupt_isr);
}

// Funktion main()
void main(void)
{
  uint16_t turns;

  setup();
  ext_interrupt_enable();

  while(1)                      // Endlosschleife loop()
  {
    lcd_setcursor(1,8);
    lcd_int(counts);
    turns = counts / m;
    lcd_setcursor(2,8);
    lcd_int(turns);

    if (bit_read(Taster,B1_SW) == 0) counts = 0;
  }

  void ext_interrupt_isr(void)
  {
    if (bit_read(Sensor,B1_B) == 1)
    {
      if (counts != 0) counts--;
    }
    else
    {
      if (counts < MAXCOUNTS ) counts++;
    }
  }
}
```