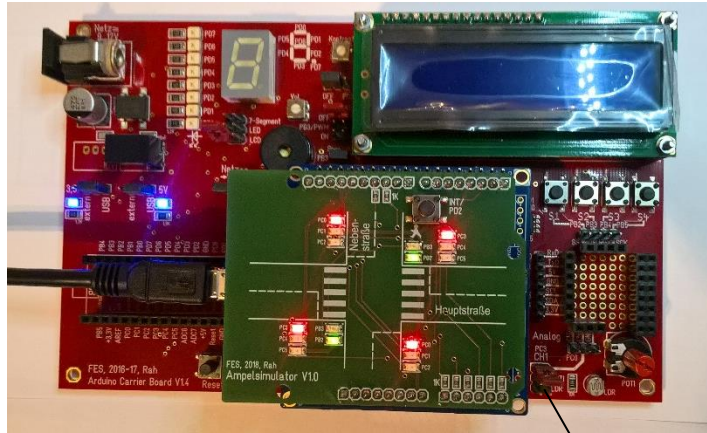
	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_1_Ampelsteuerung_Lfb.docx
	Ampelsteuerung (Lfb)	7.1.1

Projekt: Ampelsteuerungen

Entwerfen Sie verschiedenen Ampelsteuerungen unter Einsatz der Funktionen der Technischen Richtlinie.



Anschlussbelegung

PORTC 5:	Grün H(auptstraße)
PORTC 4:	Gelb H
PORTC 3:	Rot H
PORTC 2:	Grün N(ebenstraße)
PORTC 1:	Gelb N
PORTC 0:	Rot N
PORTB 3:	Rot Fussgänger
PORTB 2:	Grün Fussgänger
PORTD 2:	Anforderung Fussgänger
Bei Interrupt:	Int / (PD2)

Jumper Analog (J3) nach oben stecken!!

Arbeitsauftrag

- Erstellen Sie ein neues FA205 Projekt. Ergänzen Sie den Programmkopf und schreiben Sie das abgebildete Ampelprogramm mit einfacher Zeitsteuerung.
Testen Sie das Programm mit dem Debugger (F5).
Setzen Sie bei Bedarf Breakpoints und kontrollieren Sie die Laufvariable i in der for-Schleife.

```
#include "controller.h"


#define Ampel _PORTC_

uint8_t phase; //8 bit Variable 0...255 lokal definiert
uint8_t Ampelwerte[] = {0x09, 0x0b, 0x0c, 0x0a, 0x09, 0x19, 0x21, 0x11};
//Ampelwerte
uint16_t zeit[] = {2000, 800, 3000, 400, 1000, 800, 5000, 400};
//Zeitwerte

void setup ( void )
{ // Initialisierungen
  byte_init(Ampel,OUT);
}

int main(void)
{
  setup();

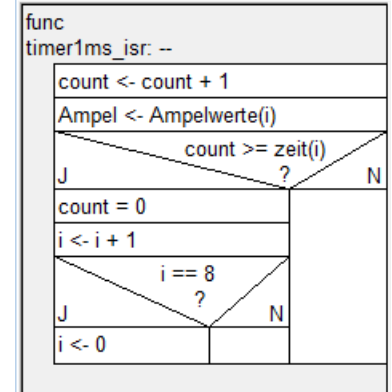
  while (1) // Endlosschleife loop()
  {
    for (phase = 0; phase < 8; ++phase) // Zähler 0 bis 7
    {
      byte_write(Ampel,Ampelwerte[phase]);
      delay_ms(zeit[phase]);
    }
  }
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_1_Ampelsteuerung_Lfb.docx
	Ampelsteuerung (Lfb)	7.1.2

2. Die Ampelphasen sollen jetzt mit Timer-Interrupt gesteuert sein. Dazu werden die FA205-Funktionen **timer1ms** verwendet. Die Funktionmalität wird nun aus der Funktion main() in die Interrupt-ISR **timer1ms_isr()** ausgelagert. Um die verschiedenen Zeiten zu generieren ist eine zusätzliche Variable erforderlich, die die Interruptaufrufe zählt. Diese Variable sollte global und als volatile deklariert werden, da Variablen in Interruptroutinen vom Compiler sonst wegoptimiert werden können:

```
volatile uint16_t count;
```

Erstellen Sie den Code zum angegebenen Struktogramm.



3. Die Ampel soll um eine Fussgänger-Anforderung ergänzt werden. Der Taster wird gepollt. Analysieren Sie das Programm und ergänzen Sie die Initialisierung und die Fussgänger-Phase:
1. Ampeln schalten auf rot
 2. Nach 2s schaltet die Fussgängerampel auf grün
 3. Nach 6s schaltet die Fussgängerampel auf rot
 4. Nach 2s geht die Ampel wieder in den normalen Betrieb

```

#include "controller.h"

#define Ampel    _PORTC_
#define Fuss     _PORTB_
#define gruen    2
#define rot      3
#define On       1
#define Off      0
#define Taster   _PORTD_
#define PD2      2

uint8_t Ampelwerte[] = {0x09, 0x0b, 0x0c, 0x0a, 0x09, 0x19, 0x21, 0x11}; //Ampelwerte
uint16_t zeit[]      = {2000, 400, 3000, 400, 1000, 400, 5000, 400}; //Zeitwerte
volatile uint8_t phase=0;
volatile uint16_t count;
volatile uint8_t status = 0;


void setup(void)
{
    // Ergänzen Sie die Initialisierung
}

void main (void)                                // Beginn Hauptprogramm
{
    setup();
    while (1)                                    // Beginn Endlosschleife
    {
        bit_toggle(Taster,PD2,&status);
    }
}

void timer1ms_isr(void)
{
    count++; // Diese Variable wird jedesmal um 1 erhöht ->jede msec
    byte_write(Ampel,Ampelwerte[phase]);

    if(count>=zeit[phase]) // Dieser Programmteil wird ausgeführt: 1msec x arraywert von zeit
    {
        count=0; phase++;
        if( phase==8 )
        {
            if( status ) //Wenn Interrupt aufgetreten und Zyklus durchgelaufen!
            {
                // Ergänzen Sie hier den Fussgänger-Zyklus
                status = 0;
            }
            phase = 0;
        }
    }
}


```

 Friedrich-Ebert- Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_1_Ampelsteuerung_LfB.docx
	Ampelsteuerung (Lfb)	7.1.3

4. Die Fussgänger-Anforderung erfolgt nun durch Interrupt. Um eine fallende Flanke an Int/ zu erzeugen, muss der Pull-Up am Interrupt-Eingang aktiviert werden:

```
bit_init(_PORTD_,2,IN);           // aktiviert den Pull-Up und sorgt für H-Pegel an Int/
```

Analysieren Sie das Programm (Lösung Aufgabe 4).

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_1_Ampelsteuerung_Lfb.docx
	Ampelsteuerung (Lfb)	7.1.4

Lösung zu Aufgabe 2)

```
#include "controller.h"

#define Ampel _PORTC_

uint8_t Ampelwerte[] = {0x09, 0x0b, 0x0c, 0x0a, 0x09, 0x19, 0x21, 0x11}; //Ampelwerte
uint16_t zeit[] = {2000, 800, 3000, 400, 1000, 800, 5000, 400}; //Zeitwerte

uint8_t phase=0;
volatile uint16_t count;


void setup(void)
{
    byte_init(Ampel,OUT);

    timer1ms_init( timer1ms_isr ); // Timer initialisieren
    timer1ms_enable(); // Timer freigeben
}

void main (void) //Beginn Hauptprogramm
{
    setup();
    while (1) //Beginn Endlosschleife
    {
        //nichts zu tun !!!
    }
}

//*****
void timer1ms_isr(void)
{
    count++; // Diese Variable wird jedesmal um 1 erhöht ->jede msec
    byte_write(Ampel,Ampelwerte[phase]);

    if(count>=zeit[phase]) // Dieser Programmteil wird ausgeführt: 1msec x arraywert von zeit
    {
        count=0; phase++;
        if( phase==8 ) phase = 0;
    }
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_L_Ampelsteuerung_Lfb.docx
	Ampelsteuerung (Lfb)	7.1.5

Lösung Aufgabe 3) Ampel mit Fussgänger-Anforderung per Polling

```
#include "controller.h"

#define Ampel    _PORTC_
#define Fuss     _PORTB_
#define gruen    2
#define rot      3
#define On       1
#define Off      0
#define Taster   _PORTD_
#define PD2      2

uint8_t Ampelwerte[] = {0x09, 0x0b, 0x0c, 0x0a, 0x09, 0x19, 0x21, 0x11}; //Ampelwerte
uint16_t zeit[]      = {2000, 400, 3000, 400, 1000, 400, 5000, 400}; //Zeitwerte

volatile uint8_t phase=0;
volatile uint16_t count;
volatile uint8_t status = 0;


void setup(void)
{
  byte_init(Ampel,OUT);
  bit_init(Fuss,gruen,OUT);
  bit_init(Fuss,rot,OUT);
  bit_write(Fuss,gruen,Off);
  bit_write(Fuss,rot,On);           // Fussgängerampel rot
  bit_init(Taster,PD2,IN);

  timer1ms_init( timer1ms_isr );    // Timer initialisieren
  timer1ms_enable();                // Timer freigeben
}

void main (void)                    // Beginn Hauptprogramm
{
  setup();
  while (1)                          // Beginn Endlosschleife
  {
    bit_toggle(Taster,PD2,&status);
  }
}

//*****
void timer1ms_isr(void)
{
  count++;                          // Diese Variable wird jedesmal um 1 erhöht ->jede msec
  byte_write(Ampel,Ampelwerte[phase]);

  if(count>=zeit[phase]) // Dieser Programmteil wird ausgeführt: 1msec x arraywert von zeit
  {
    count=0; phase++;
    if( phase==8 )
    {
      if( status )                //Wenn Interrupt aufgetreten und Zyklus durchgelaufen!
      {
        byte_write(Ampel,0x09);    //Ampel auf rot
        delay_ms(1000);
        bit_write(Fuss,gruen,On);
        bit_write(Fuss,rot,Off);    // Fussgängerampel grün
        delay_ms(5000);
        bit_write(Fuss,gruen,Off);
        bit_write(Fuss,rot,On);     // Fussgängerampel rot
        delay_ms(2000);
        status = 0;
      }
      phase = 0;
    }
  }
}
```

 Friedrich-Ebert-Schule Esslingen FES	Mikrocontroller-Labor	Name: Rahm Datum: 05.02.2017 7_1_Ampelsteuerung_Lfb.docx
	Ampelsteuerung (Lfb)	7.1.6

Lösung Aufgabe 4: Ampel mit Fussgänger-Anforderung per Interrupt

```
#include "controller.h"

#define Ampel    _PORTC_
#define Fuss     _PORTB_
#define gruen    2
#define rot      3

#define On       1
#define Off      0
uint8_t Ampelwerte[] = {0x09, 0x0b, 0x0c, 0x0a, 0x09, 0x19, 0x21, 0x11}; //Ampelwerte
uint16_t zeit[]      = {2000, 800, 3000, 400, 1000, 800, 5000, 400}; //Zeitwerte
volatile uint8_t phase=0, int_status; //volatile, sonst werden die Variablen rausoptimiert??
uint16_t count=0;

void setup(void)
{
  byte_init(Ampel,OUT);
  bit_init(Fuss,gruen,OUT);
  bit_init(Fuss,rot,OUT); //
  bit_init(_PORTD_,2,IN); // Pull-Up für Interrupteingang aktivieren!

  bit_write(Fuss,gruen,Off);
  bit_write(Fuss,rot,On); // Fussgängerampel rot

  timer1ms_init( timer1ms_isr );
  timer1ms_enable();
  ext_interrupt_init( ext_interrupt_isr );
  ext_interrupt_enable();
}

void main (void) //Beginn Hauptprogramm
{
  setup();
  while (1) //endlos
  {
    if( int_status & !phase ) //Wenn Interrupt aufgetreten und Zyklus durchgelaufen!
    {
      timer1ms_disable(); //Timer sperren sonst läuft Ampel normal weiter
      int_status = 0;
      byte_write(Ampel,0x09); //Ampel auf rot
      delay_ms(2000);
      bit_write(Fuss,gruen,On);
      bit_write(Fuss,rot,Off); // Fussgängerampel grün
      delay_ms(6000);
      bit_write(Fuss,gruen,Off);
      bit_write(Fuss,rot,On); // Fussgängerampel rot
      delay_ms(2000);
      timer1ms_enable();
    }
  }
}

void timer1ms_isr(void)
{
  count++; // Diese Variable wird jedesmal um 1 erhöht ->jede msec
  byte_write(Ampel,Ampelwerte[phase]);

  if(count>=zeit[phase]) // Dieser Programmteil wird ausgeführt: 1msec x arraywert von zeit
  {
    count=0;
    phase=phase+1;
    if( phase==8 ) phase=0;
  }
}

void ext_interrupt_isr(void)
{
  int_status = 1; // Merker für Interrupt
}
```