

Hauptprüfung 18/19 A1

Sortieranlage

Veröffentlicht unter:

<https://os.mbed.com/users/jack1930/code/HP1819A1/>

Import Programm: Suchbegriffe GSOE HP1819A1



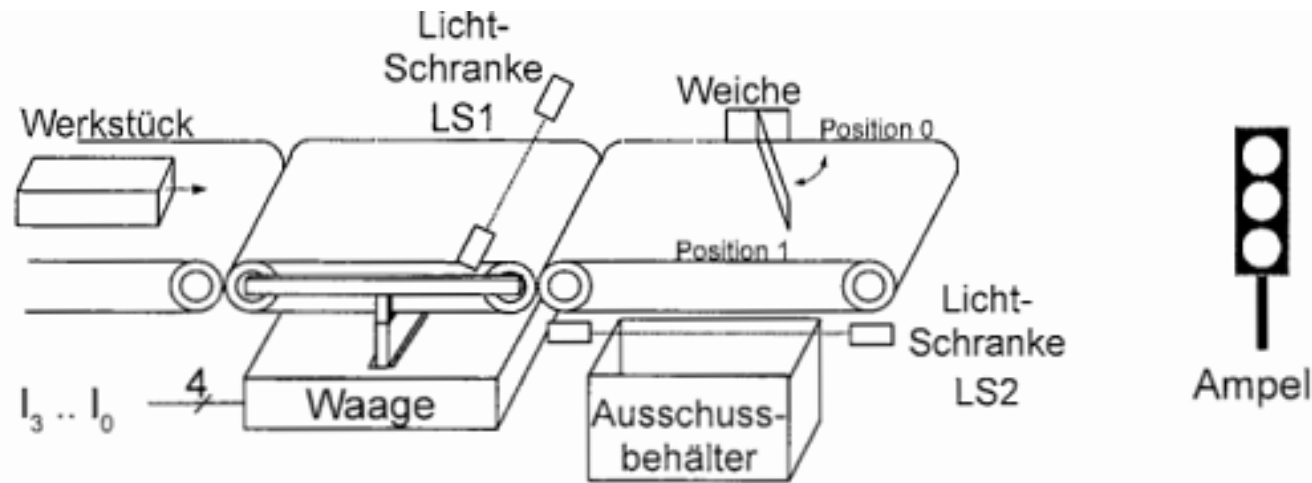
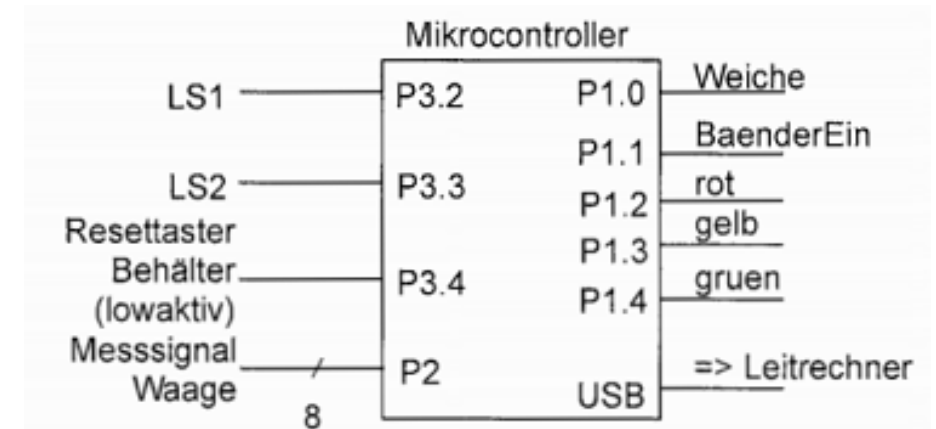


Abb. 1: Technologieschema



Über Fließbänder werden Werkstücke über eine Digitalwaage transportiert, dabei gewogen und anschließend auf einem weiteren Transportband mittels einer Weiche sortiert. Die Waage gibt einen Gewichtsmesswert in 1 kg-Schritten, von 0 bis 15 kg binär codiert, an den Signalen $I_3 \dots I_0$ aus. Das Mindestgewicht wird mit den Schaltern $S_3 \dots S_0$ binär codiert eingestellt (siehe Abb. 2). Die Lichtschranke $LS1$ meldet mit fallender Flanke, dass ein Werkstück vollständig auf der Waage liegt, das Ergebnis der Wägung liegt dann bereits vor. Die Weiche wird daraufhin in Abhängigkeit vom Werkstückgewicht eingestellt. Ist das Werkstückgewicht (Istwert $I_3 \dots I_0$) kleiner als das Mindestgewicht (Signal $X = 1$), dann wird die Weiche mit dem Signal $Weiche=1$ in Position 1 gebracht und das Werkstück in den Ausschussbehälter befördert. Anderenfalls (Signal $X = 0$) wird die Weiche mit dem Signal $Weiche=0$ in Position 0 gebracht und das Werkstück weiter transportiert. Werkstücke die in den Ausschussbehälter fallen werden durch Lichtschranke $LS2$ erfasst. Die Lichtschranken $LS1$ und $LS2$ liefern ein „0“-Signal, wenn sie ein Werkstück erkennen.



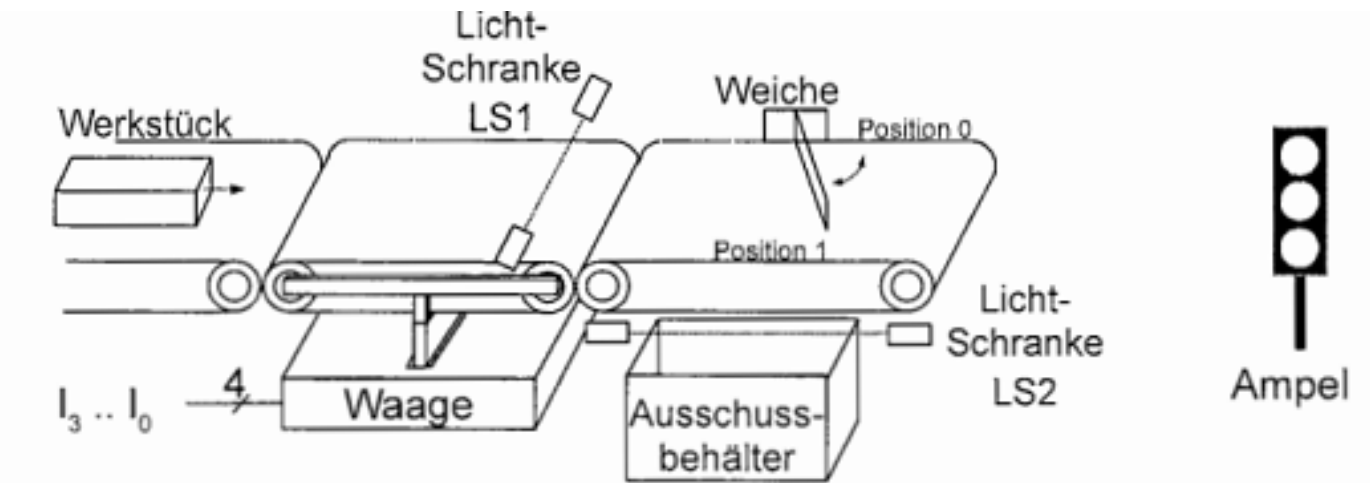
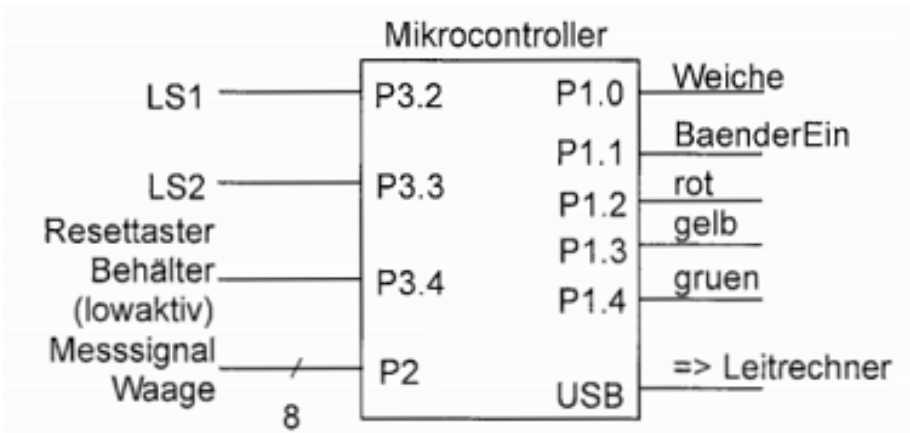


Abb. 1: Technologieschema



Anzahl der Teile im Behälter	Ampel	BaenderEin
0	Aus	1
1	grün	1
2	gelb	1
3	gelb und rot	1
4	rot	0

Die Signale:
gruen, gelb, rot und
BaenderEin
sind highaktiv

Tabelle 1:
Ausschussbehälter-Anzeige



Hauptprüfung 18/19 A1

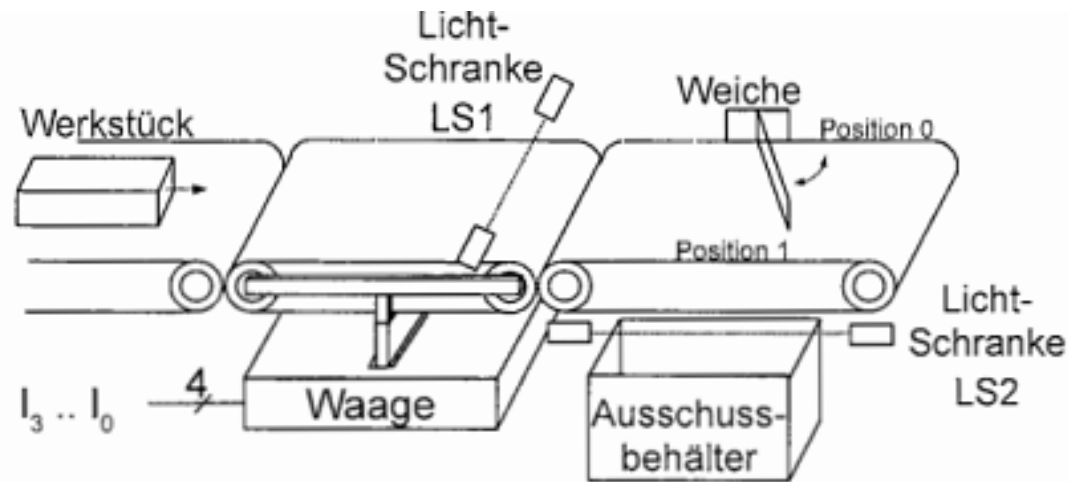
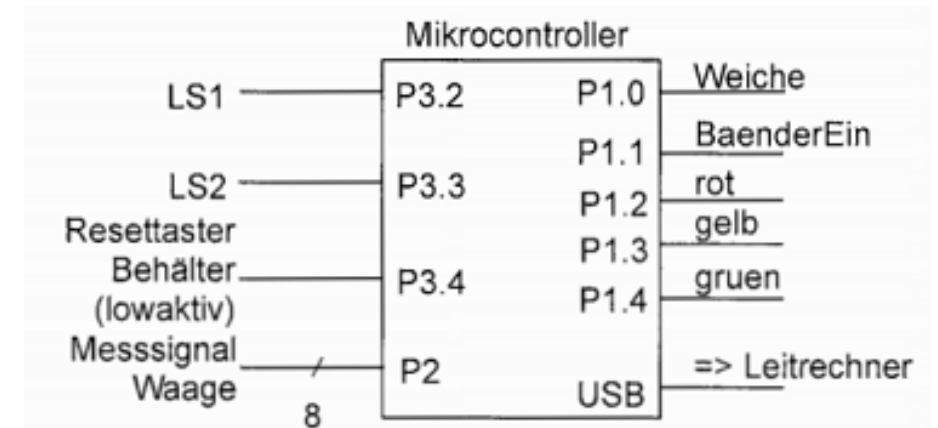


Abb. 1: Technologieschema



Ampel



```

InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b11111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
Maschinenzyklus: 1/32000000 s
    
```



1.2.1 Schreiben Sie das Unterprogramm *init/SR* für die Initialisierung der externen Interrupts.

```
InterruptIn ls1(PA_1);  
InterruptIn ls2(PA_6);  
DigitalIn reset(PA_10);  
PortIn waage(PortB,0xFF);  
PortOut ausgaenge(PortC,0b111111);  
DigitalOut Weiche(PC_0);  
DigitalOut BaenderEin(PC_1);  
DigitalOut rot(PC_2);  
DigitalOut gelb(PC_3);  
DigitalOut gruen(PC_4);
```



1.2.1 Schreiben Sie das Unterprogramm *initISR* für die Initialisierung der externen Interrupts.

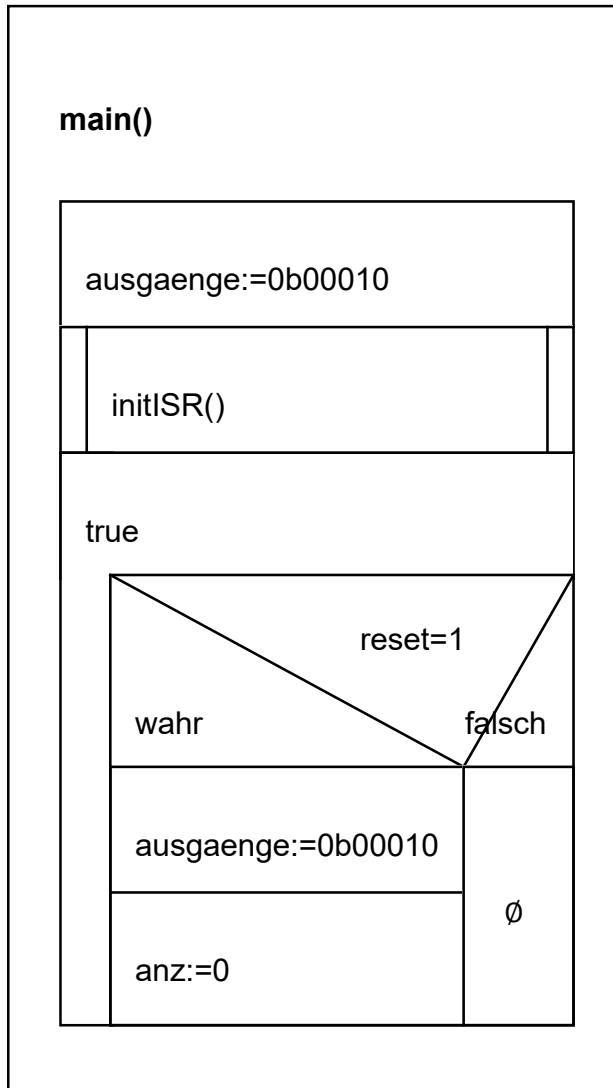
```
void initISR()
{
    ls1.fall(&isrLS1);
    ls1.enable_irq();
    ls2.fall(&isrLS1);
    ls2.enable_irq();
    __enable_irq();
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.2 Stellen Sie den Algorithmus des Hauptprogramms in einem Programmablaufplan (PAP) oder einem Struktogramm dar. (*initISR* verwenden)



- Hauptprogramm initialisiert die Ampel (*Aus*), die Transportbänder (*Ein*), die Weiche in Position 0, die externen Interrupts (*UP initISR*) und prüft zyklisch den *Behälter-Resettaster*:
- Bei Betätigung des *Behälter-Resettasters*: Ampel aus-, Transportbänder einschalten und Zählvariablen bzw. -register auf 0 setzen.

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b11111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



1.2.3 Entwickeln Sie den Algorithmus der ISR zur Auswertung der Lichtschranke LS2 und stellen Sie diesen in einem Programmablaufplan (PAP) oder einem Struktogramm dar.

isrLS2()

lokale Variable char ampel[5]={0b00010, 0b10010, 0b01010, 0b01110, 00b10000}

anz:=anz+1

ausgaenge:=ampel[anz];

3

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b11111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```

- LS2 erkennt Werkstück (LS2: 1 → 0):
Ampel weiter schalten (siehe Tabelle 1) und Bänder stoppen wenn die Anzahl der Ausschussteile im Behälter die Zahl 4 erreicht.




```
InterruptIn ls1(PA_1);  
InterruptIn ls2(PA_6);  
DigitalIn reset(PA_10);  
PortIn waage(PortB,0xFF);  
PortOut ausgaenge(PortC,0b11111);  
DigitalOut Weiche(PC_0);  
DigitalOut BaenderEin(PC_1);  
DigitalOut rot(PC_2);  
DigitalOut gelb(PC_3);  
DigitalOut gruen(PC_4);
```

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

- *LS1* erkennt Werkstück (*LS1*: $1 \rightarrow 0$):
Transportbänder stoppen, 1 Sekunde warten, danach entsprechend dem Gewicht die Weiche einstellen und die Transportbänder wieder starten.



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;           //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;              //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;             //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;    //Prescaler für 1ms
    TIM6->CNT=0;        //Counter mit 0 starten
    TIM6->ARR=999;      //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;         //Überlaufflag zurücksetzen
    TIM6->CR1=1;        //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0); //Warten auf Überlauf
    TIM6->CR1=0;        //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999; //Prescaler für 1ms
    TIM6->CNT=0; //Counter mit 0 starten
    TIM6->ARR=999; //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0; //Überlaufflag zurücksetzen
    TIM6->CR1=1; //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0); //Warten auf Überlauf
    TIM6->CR1=0; //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999; //Prescaler für 1ms
    TIM6->CNT=0; //Counter mit 0 starten
    TIM6->ARR=999; //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0; //Überlaufflag zurücksetzen
    TIM6->CR1=1; //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0); //Warten auf Überlauf
    TIM6->CR1=0; //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;           //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;             //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;             //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;          //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;              //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;            //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;           //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;              //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;             //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;           //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;              //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;             //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



Hauptprüfung 18/19 A1

1.2.4 Entwerfen Sie den Programmteil für *LS1* in Assembler oder C. Für die Wartezeit ist die Verwendung eines Timers gefordert.

```
void isrLS1()
{
    BaenderEin=0;
    warte1s();
    if (sollwert > waage)
    {
        Weiche=1;
    }
    else
    {
        Weiche=0;
    }
    BaenderEin=1;
}
```

```
void warte1s()
{
    RCC->APB1ENR |= 0b10000; //TIM6 mit Takt versorgen
    TIM6->PSC=31999;        //Prescaler für 1ms
    TIM6->CNT=0;             //Counter mit 0 starten
    TIM6->ARR=999;           //Autoreload für 1s (1000ms Zählwert von 0-999)
    TIM6->SR=0;              //Überlaufflag zurücksetzen
    TIM6->CR1=1;             //setzt CEN auf 1 => startet den Timer
    while (TIM6->SR==0);     //Warten auf Überlauf
    TIM6->CR1=0;             //setzt CEN auf 0 => stoppt den Timer
}
```

```
InterruptIn ls1(PA_1);
InterruptIn ls2(PA_6);
DigitalIn reset(PA_10);
PortIn waage(PortB,0xFF);
PortOut ausgaenge(PortC,0b111111);
DigitalOut Weiche(PC_0);
DigitalOut BaenderEin(PC_1);
DigitalOut rot(PC_2);
DigitalOut gelb(PC_3);
DigitalOut gruen(PC_4);
```



1.2.5 Mikrocontroller-Grundlagen: Programcounter PC (Befehlszähler) Erläutern Sie Aufbau und Inhalt des Programcounter. Beschreiben Sie, wie sich Sprungbefehle auf den Inhalt des Programcounter auswirken.

- Der Programcounter beinhaltet immer die Adresse im Befehls-ROM, der Instruktion, die als nächstes ausgeführt werden soll.
- Beim Programcounter handelt es sich um das 32Bit-Register R15.
- Bei einem Sprungbefehl wird in den Programcounter die Adresse des Sprungziels eingetragen

```
InterruptIn ls1(PA_1);  
InterruptIn ls2(PA_6);  
DigitalIn reset(PA_10);  
PortIn waage(PortB,0xFF);  
PortOut ausgaenge(PortC,0b111111);  
DigitalOut Weiche(PC_0);  
DigitalOut BaenderEin(PC_1);  
DigitalOut rot(PC_2);  
DigitalOut gelb(PC_3);  
DigitalOut gruen(PC_4);
```

