

# Versuch V1

C752 Digitaltechnik

Stand: 4. November 2023

*Tom Mohr  
Martin Ohmeyer*

# Inhaltsverzeichnis

<b>1 Aufgabe 3</b>	<b>1</b>
1.1 Aufgabe 3.1 . . . . .	1
<b>2 Aufgabe 6</b>	<b>2</b>
2.1 Aufgabe 6.1 . . . . .	2
2.2 Aufgabe 6.2 . . . . .	2
<b>3 Aufgabe 7</b>	<b>3</b>
3.1 Aufgabe 7.1 . . . . .	3
3.2 Aufgabe 7.2 . . . . .	3
3.3 Aufgabe 7.3 . . . . .	3
3.4 Aufgabe 7.4 . . . . .	3
<b>4 Aufgabe 9</b>	<b>4</b>
4.1 Aufgabe 9.1 . . . . .	4
4.2 Aufgabe 9.2 . . . . .	4
4.3 Aufgabe 9.3 . . . . .	4
4.4 Aufgabe 9.4 . . . . .	4
<b>5 Aufgabe 10</b>	<b>5</b>
5.1 Aufgabe 10.1 . . . . .	5
<b>6 Aufgabe 11</b>	<b>8</b>
6.1 Aufgabe 11.1 . . . . .	8
6.2 Aufgabe 11.2 . . . . .	9
6.3 Aufgabe 11.3 . . . . .	13

# 1 Aufgabe 3

## 1.1 Aufgabe 3.1

**Aufgabenstellung** Sollten Sie bei den folgenden Punkten Wissenslücken feststellen, füllen Sie diese bitte vor dem Laborversuch selbstständig auf z.B. durch YouTube Videos.

**Vorüberlegung** Es bestehen Defizite im Umgang mit Multimeter, Oszilloskop und der Logikgatter.

**Durchführung** Das Lesen der Bedienungsanleitung und ausprobieren von verschiedenen Einstellungen hilft dabei, die Geräte besser zu verstehen. Die Datenblätter zu einigen Logikgattern findet man im Internet.

**Schlussfolgerung** Der neue Informationsstand ist ausreichend, um die folgenden Aufgaben gewissenhaft zu erledigen.

## **2 Aufgabe 6**

**2.1 Aufgabe 6.1**

**2.2 Aufgabe 6.2**

# **3 Aufgabe 7**

**3.1 Aufgabe 7.1**

**3.2 Aufgabe 7.2**

**3.3 Aufgabe 7.3**

**3.4 Aufgabe 7.4**

# **4 Aufgabe 9**

**4.1 Aufgabe 9.1**

**4.2 Aufgabe 9.2**

**4.3 Aufgabe 9.3**

**4.4 Aufgabe 9.4**

# 5 Aufgabe 10

## 5.1 Aufgabe 10.1

**Aufgabenstellung** Nutzen Sie das Beispiel aus Listing 1 bzw. die Beispiele aus dem B15-Git, um ein Programm zu schreiben welches zwei Betriebsmodi besitzt. In Zustand A sollen die digitalen Eingänge invertiert auf die digitalen Ausgänge weitergereicht werden. In Zustand B soll auf den digitalen Ausgängen das Lichtmuster aus dem Film Knightrider dargestellt werden. Unterscheiden Sie beide Betriebsmodi, indem Sie einen der DIP-Schalter auswerten.

**Vorüberlegung** Das B15-Board verfügt an allen BA und BE Steckleisten über 8 Anschlüsse. LEDs hinter den Anschlussbuchsen stellen die aktuelle Dezimalzahl in Binär dar. Rechnet man  $2^8$  ergibt dies 256, folglich können hier Zahlen von Null bis 255 Ein- sowie Ausgegeben werden. Um das Eingangssignal zu Invertieren, kann man daher einfach den Eingang von 255 subtrahieren. Das darzustellende Muster aus dem angegebenen Film ist ein einfaches hin- und herwandern einer einzelnen aktiven LED.

**Durchführung** In einer endlos laufenden Schleife soll der Wert des DIP-Schalters (in Abbildung 5.1 oben links), welcher Teil des B15-Boards ist, abgefragt und die entsprechenden Zustände ausgeführt werden. Steht der Schalter auf 0, soll das Programm *Zustand A* einmal ausgeführt. Sollte der Wert 1 sein, wird *Zustand B* einmal durchlaufen. In Zustand A wird der Wert des Blocks BE-0 ausgelesen, von 255 subtrahiert und das Ergebnis auf BA-0 geschrieben, wie in Abbildung 5.2 auf der nächsten Seite gezeigt. In Zustand B werden zwei Zählerschleifen nacheinander ausgeführt, welche den Dezimalwert der aktuellen LED halbieren bzw. verdoppeln und ihn mit einer leichten Verzögerung, die dem Knightrider Muster ähnelt, auf den Ausgang BA-0 schreibt. Der Quelltext kann nachfolgend entnommen werden.

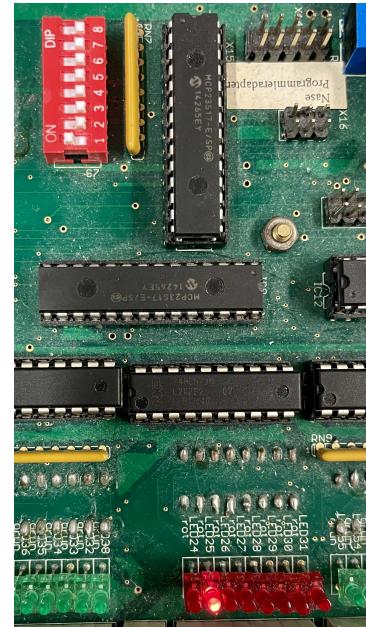


Abb. 5.1: DIP-Schalter

## 5 Aufgabe 10

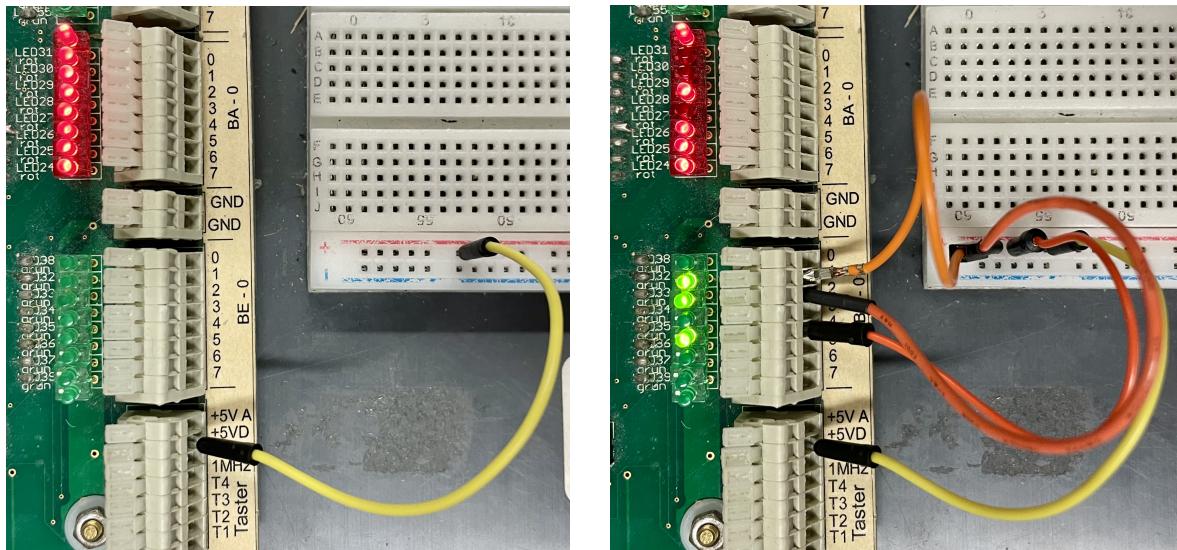


Abb. 5.2: Zustand A



Abb. 5.3: Zustand B

## 5 Aufgabe 10

**Schlussfolgerung** Das B15-Board verhält sich wie in der Aufgabenstellung gefordert und reagiert schnell. Sollte jedoch ein Wert größer als 1 am DIP-Schalter eingestellt sein, wird nichts ausgeführt. Der Versuch ist erfolgreich.

```
1 #include <b15f/b15f.h>
2
3 int main()
4 {
5     // Verbindung zum B15-Board aufbauen
6     B15F &drv = B15F::getInstance();
7
8     int mode = 0;
9     while (true)
10    {
11        // DIP Schalter auslesen um aktiven Zustand zu ermitteln
12        mode = drv.readDipSwitch();
13
14        if (mode == 0) // Zustand A
15        {
16            // Digitalen Eingang auf digitalen Ausgang invertieren
17            drv.digitalWrite0(255 - (int)drv.digitalRead0());
18        }
19        else if (mode == 1) // Zustand B
20        {
21            // Lauflicht in Richtung 1
22            for (int output = 64; output > 1; output /= 2)
23            {
24                drv.digitalWrite0(output);
25                drv.delay_ms(150);
26            }
27            // Lauflicht in Richtig 2
28            for (int output = 1; output <= 128; output *= 2)
29            {
30                drv.digitalWrite0(output);
31                drv.delay_ms(150);
32            }
33        }
34    }
35 }
36 }
```

# 6 Aufgabe 11

## 6.1 Aufgabe 11.1

**Aufgabenstellung** Wählen Sie ein beliebiges Logikgatter, verbinden Sie dessen Eingänge mit dem B15-Board. Nutzen Sie das Monitor-Programm um die Wahrheitswertetabelle Ihres Logikgatters zu erstellen.

**Vorüberlegung** Ausgewählt wurde ein AND Gatter vom Typ *DL008D*. Der Ausgang sollte erst durchgeschaltet werden, wenn beide Eingänge eingeschaltet sind.

**Durchführung** Das Gatter wird mit 5V Betriebsspannung über Pin 14 versorgt und über Pin 7 geerdet. Die Pins 1 und 2 sind Eingänge, dessen Ergebnis wird auf Pin 3 ausgegeben. Pin 1 wird mit Pin 0 und Pin 2 mit Pin 1 des Feldes BA-0 vom B15-Board verbunden. Die Ausgabe des Gatters wird auf BE-0 Pin 0 wiedergegeben. Der Aufbau kann Abbildung 6.1 entnommen werden. Zum testen dieser Schaltung wird der Wert von BA-0 mit den Binärkombinationen 00, 01, 10 und 11 nacheinander über das Monitor-Programm eingestellt. Die daraus entstandene Wahrheitswertetabelle ist in Abbildung 6.2 dargestellt.

**Schlussfolgerung** Das Ergebnis entspricht dem Erwartungswert. Der Versuch ist erfolgreich.

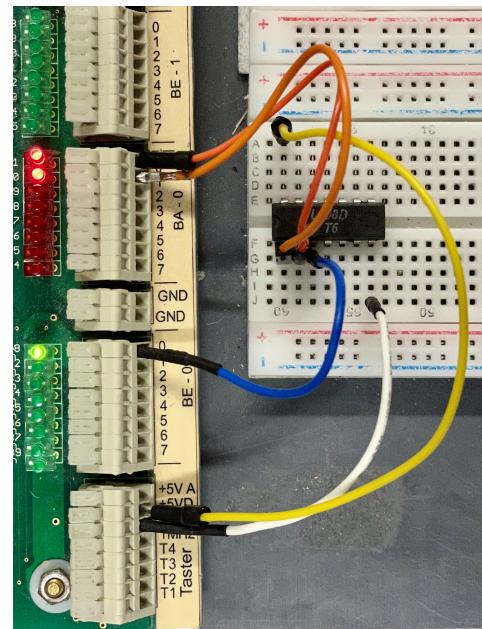


Abb. 6.1: Aufbau des verwendeten UND-Gatters

$E_2$	$E_1$	$A$
0	0	0
0	1	0
1	0	0
1	1	1

Abb. 6.2: Wahrheitswertetabelle der UND-Verknüpfung

## 6 Aufgabe 11

### 6.2 Aufgabe 11.2

**Aufgabenstellung** Wählen Sie mindestens 3 beliebige Logikgatter aus. Erstellen Sie dar- aus eine Logikschaltung die mindestens 4 Eingangsvariablen hat. Entwickeln Sie mit Hilfe des B15 ein Programm welches automatisch die Wahrheitswertetabelle des Logikgatters ausgibt. Überprüfen Sie die Ausgabe.

**Vorüberlegung** Zur verfügbaren Auswahl ständen AND-, NAND-, NOR- und XOR-Gatter. XOR wurde schnell ausgeschlossen, da die gefundene Dokumentation zur Bedienung unzureichend für den Betrieb ist. Nach einigen Tests für eine Schaltung, die ein durchwachsenes und nicht einseitiges Ergebnis liefert, ergaben sich Probleme bei der Verwendung des NAND-Gatters. Eine passende Schaltung mit zwei AND- und einer NOR-Verknüpfung wurde gefunden. Das Programm, welches die Wahrheitswertetabelle ausgeben soll, sollte möglichst universell Einsetzbar sein. Daher sollte es eine variable Größe von Eingängen verarbeiten können. Da eine Eingabe von nötig ist, um dies umzusetzen, darf diese nur Zahlen im Bereich von 1 bis 255 akzeptieren. Das Ergebnis sollte gut lesbar in Tabellenform ausgegeben werden und Abbildung 6.3 entsprechen.

**Durchführung** Die Gatter AND (*Typ DL008D*) und NOR (*Typ DL002D*) werden auf dem Steckbrett wie in Abbildung 6.4 auf der nächsten Seite gezeigt positioniert und jeweils über die Pins 14 mit 5V Betriebsspannung versorgt und über Pin 7 geerdet. Der Pin 1 des AND-Gatter wird mit BA-0 Pin 0 und Pin 2 mit BA-0 Pin 1 des B15-Boards verbunden. Das Ergebnis beider Pins wird über Pin 3 mit Pin 3 des NOR-Gatters verbunden. Eine weitere UND-Verknüpfung wird mit Pin 13 über Pin 3 und Pin 12 des AND-Gatters über Pin 4 der Buchsen BA-0 des B15-Boards verbunden. Das Ergebnis dieser Verknüpfung wird von Pin 11 des AND-Gatters zu Pin 2 des NOR-Gatters geleitet. Die benötigten Eingänge zur Berechnung des NOR sind dadurch belegt und das Ergebnis dieser wird über Pin 1 des NOR-Gatters mit Pin 0 von BE-0 des B15-Boards verbunden. Die Schaltung ist nun Vollständig und kann ebenfalls Abbildung 6.4 auf der nächsten Seite entnommen werden. Folgende Formel beschreibt den Aufbau:

$E_4$	$E_3$	$E_2$	$E_1$	$A$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Abb. 6.3: Wahrheitswertetabelle der Schaltung

$$\neg((a \wedge b) \vee (c \wedge d))$$

6 Aufgabe 11

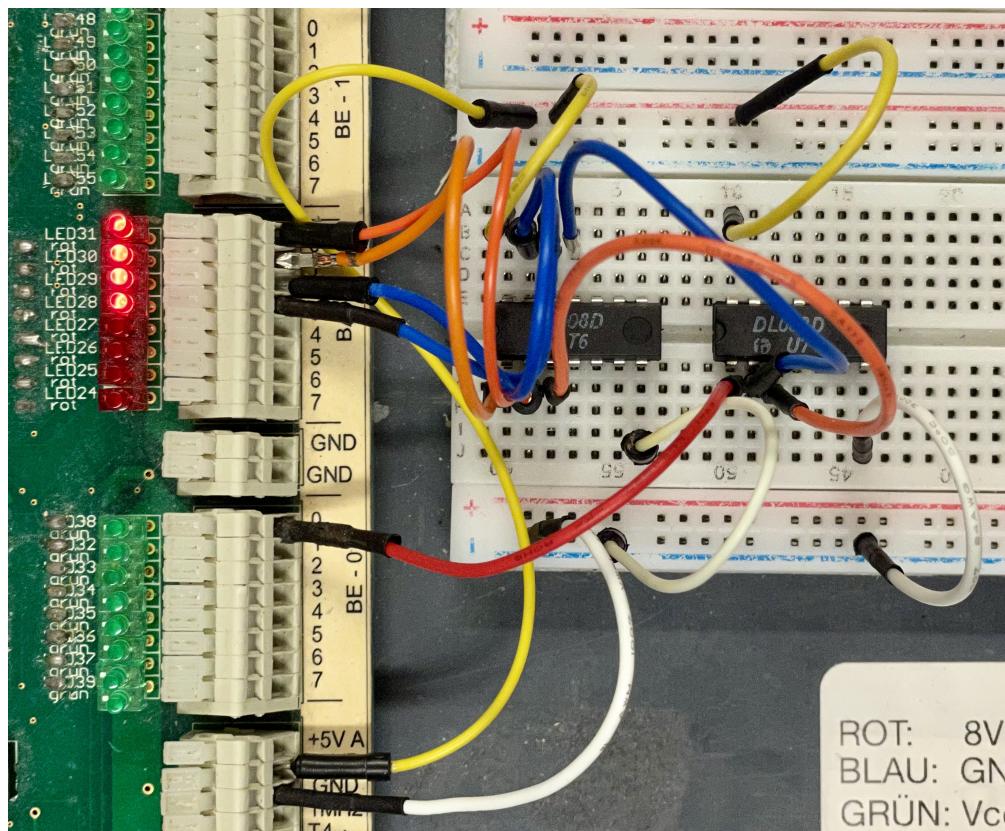


Abb. 6.4: Aufbau der Logischen Schaltung

## 6 Aufgabe 11

Das Programm baut eine Verbindung zum B15-Board auf, fragt den Benutzer nach der Testgröße und durchläuft alle Zahlen innerhalb dieser in aufsteigender Reihenfolge. Der Quelltext kann nachfolgend gelesen werden.

```
1 #include <iostream>
2 #include <string>
3 #include <b15f/b15f.h>
4
5 using namespace std;
6
7 /// @brief Konvertiert eine Zahl von Dezimal in Binär
8 /// @param n Die natürliche Zahl, die konvertiert werden soll.
9 /// @return den Binären Wert von `n`
10 string toBinary(int n)
11 {
12     string r;
13     while (n != 0)
14     {
15         r = (n % 2 == 0 ? "0" : "1") + r;
16         n /= 2;
17     }
18     return r;
19 }
20
21 /// @brief Fordert die Eingabe einer Zahl an und prüft diese auf
22 //→ Kompatibilität
22 /// @param errorOnLastRun `true`, wenn der letzte Aufruf nicht
23 //→ erfolgreich war.
23 /// @return Die korrekte Eingabe als Zahl
24 int inputRequest(bool errorOnLastRun = false)
25 {
26     // Erzeugt eine Fehlermeldung bei unpassender Eingabe
27     if (errorOnLastRun)
28     {
29         cout << "\nDie Eingabe muss eine Ganzzahl sein, welche zur
29 //→ Reihe der 2er-Potenzen gehört und nicht größer als 256
29 //→ ist." << endl;
30         cin.clear();
31         cin.ignore(10000, '\n');
32     }
33
34     cout << "Anzahl der maximalen Zeilen: ";
35     int decimalInput = 0;
36
37     // Eingabe auf Zahl prüfen
38     if (!(cin >> decimalInput))
39     {
40         // Eingabe erneut abfragen
41         return inputRequest(true);
42     }
43
44     int i = 1;
```

## 6 Aufgabe 11

```
45 // Fehlerhafte Eingabe mit nächst-größerer Zahl korrigieren
46 while (decimalInput <= 256 and decimalInput > i)
47 {
48     i *= 2;
49 }
50
51 // Prüft die eingegebene Zahl auf Gültigkeit
52 return decimalInput == i ? decimalInput : inputRequest(true);
53 }
54
55 int main()
56 {
57     // Verbindung zum B15-Board aufbauen
58     B15F &drv = B15F::getInstance();
59
60     // Die Anzahl der zu testenden Ausgänge abfragen
61     int max = inputRequest();
62     cout << "\n\n";
63
64     for (int i = 0; i < max; i++)
65     {
66         // Aktuellen Testausgang konvertieren
67         string bin = toBinary(i);
68
69         // Führende Nullen ergänzen
70         while (bin.length() < (log(max) / log(2)))
71         {
72             bin = bin.insert(0, "0");
73         }
74
75         // Separator für die Darstellung einfügen
76         int length = bin.length() * 4 - 3;
77         for (int j = 0; j < length; j += 4)
78         {
79             bin.insert(j, " | ");
80         }
81
82         // Tabellenkopf erzeugen
83         if (i == 0)
84         {
85             string letters = " | a | b | c | d | e | f | g | h | ";
86             string header = letters.substr(0, bin.length() * 4 + 2);
87             cout << header << "| A | \n"
88                 << endl;
89         }
90
91         // Ausgang auf Eingang testen
92         drv.digitalWrite0(i);
93         int input = (int)drv.digitalRead0();
94
95         // Ergebnis ausgeben
96         cout << bin << " | " << input << " | " << endl;
```

## 6 Aufgabe 11

```
97     }
98
99     return 0;
100 }
```

**Schlussfolgerung** Das Ergebnis des Programms entspricht dem Erwartungswert. Der Versuch ist erfolgreich. Die exakte Ausgabe des Programms lautet:

```
[B15F] Verwende Adapter: /dev/ttyUSB0
[B15F] Stelle Verbindung mit Adapter her... OK
[B15F] Teste Verbindung... OK
[B15F] AVR Firmware Version: Dec 3 2019 um 13:14:09 Uhr (boardinfo.h)
Anzahl der maximalen Zeilen: 16
```

	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		0	
	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		0	
	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		1	
	0		0		0		0		0	
	0		0		0		0		0	
	0		0		0		0		0	
	0		0		0		0		0	
	0		0		0		0		0	
	0		0		0		0		0	
	0		0		0		0		0	

## 6.3 Aufgabe 11.3

**Aufgabenstellung** Räumen Sie alle Kabel und Widerstände die Sie aus dem Widerstandssortiment von der Wand genommen haben wieder zurück.

**Durchführung** Das Netzteil wird ausgeschaltet und die Kabel zur Stromversorgung abgeklemmt. Die Widerstände werden in entsprechende Box zurückgelegt und die Kabel der Schaltungen einzeln vom Steckbrett gelöst und gesammelt in die Kabelbox gelegt. Mit dem Bauteilegreifer werden die verwendeten Logikgatter einzeln vom Steckbrett gelöst und in die entsprechende Sammelbox gelegt.

**Schlussfolgerung** Der Arbeitsplatz befindet sich in einem sauberen Zustand und kann wieder verwendet werden.