Wir übertragen Daten mit Licht

Durch das Internet werden täglich Unmengen von Daten von einem Ort an den anderen transportiert. Häufig geschieht dies über Glasfasern (Abb. 1). An dem einen Ende werden Lichtsignale in die Faser eingespeist. Das Licht wandert dann durch die Faser; dabei verhindert die Totalreflexion, dass das Licht die Faser seitlich verlassen kann. Am Ende der Faser befindet sich ein lichtempfindlicher Sensor, der das Lichtsignal wieder in ein elektrisches Signal umwandelt.

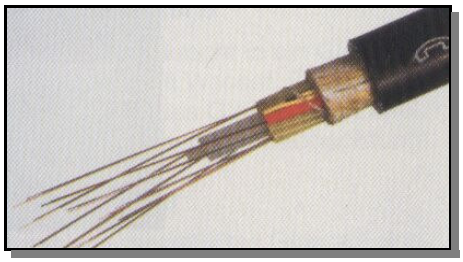

Abb. 1: Ein Glasfaserkabel

Auch mit unseren Platinen können wir eine solche Form der Datenübertragung zwischen zwei Rechner herstellen. Der Rechner, welche die Nachricht sendet, hat auf seiner Platine eine Leuchtdiode; diese wird über einen Lichtleiter mit einem Fototransistor verbunden, welcher sich auf der Platine des Empfänger-Rechners befindet. Der Fototransistor verändert seine Leitfähigkeit je nach Lichteinfall; er kann also wie unser LDR als optischer Sensor benutzt werden. Leuchtdiode und Fototransistor haben ähnliche Bauformen; an beide lässt sich unser Lichtleiter leicht mit einem Schrumpfschlauch befestigen (Abb. 2).

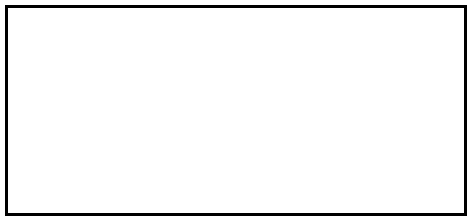

Abb. 2: LED und Fototransistor

Um nun mit dieser Anordnung Texte übertragen zu können, müssen wir Programme schreiben, welche aus einem Text passende Lichtsignale und umgekehrt aus den Lichtsignalen wieder den Text rekonstruieren können. Wie dies geht, wollen wir in den folgenden Abschnitten zeigen.

Das Prinzip der seriellen Datenübertragung

Abb. 3 macht deutlich, wie wir vorgehen wollen, um einen Text zu übertragen: Zunächst wird der Text in seine einzelnen Zeichen zerlegt: Buchstaben, Ziffern, Satzzeichen u.s.w.. Für jedes dieser Zeichen gibt es einen ASCII-Code, eine Zahl zwischen 0 und 255. Statt des Zeichens übertragen wir nun diese Zahl. Wie aber können wir dies mithilfe des Lichts bewerkstelligen? Eine Möglichkeit besteht darin, die Zahlen von 0 bis 255 durch entsprechend viele unterschiedliche Helligkeiten darzustellen. Diese analoge Übertragung zuverlässig zu realisieren, ist aber sehr schwer; denn schon unbeabsichtigte Helligkeitsschwanken von unter 1% verfälschen die Übertragung enorm: Aus einer gesendet Zahl 90 wird dann vielleicht eine empfangene Zahl 91, d.h. aus einem „Z“ wird ein „[“.

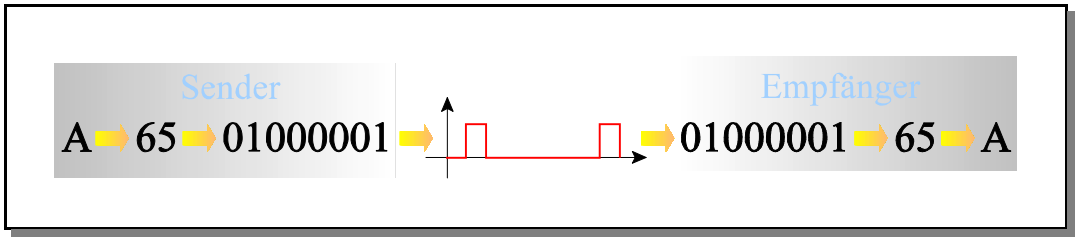

Abb. 3: Prinzip der seriellen Datenübertragung

Aus diesem Grund wird üblicherweise eine digitale Übertragung der ASCII-Codes verwendet. Dazu stellen wir die Zahl zunächst im Zweiersystem dar. Die einzelnen Ziffern einer solchen Dualzahl sind entweder 0 oder 1. Diese Ziffern werden nun durch Lichtsignale (LICHT AUS für 0, LICHT AN für 1) **seriell**, das heißt nacheinander, übertragen.

Soll z.B. die Dualzahl „01011101“ übertragen werden, so muss zunächst festgelegt werden, wann die Übertragung beginnt. Wenn keine Datenübertragung stattfindet, ist das Licht aus. Um anzuzeigen, dass eine Dualzahl (genauer: 1 Byte = 8 Bit) übertragen werden soll, wird als erstes ein 1-Signal (LICHT AN) als **Startbit** gesendet. Danach folgen in festem Zeitabstand die Zustände der einzelnen Datenbits, beginnend mit dem höherwertigen Bit 7. Den Abschluss bildet das **Stoppbit** mit dem Wert 0. Danach ist der Lichtleiter wieder frei für die nächste Übertragung. Für jedes Bit steht eine bestimmte Zeit, die **Bitzeit**, zur Verfügung.

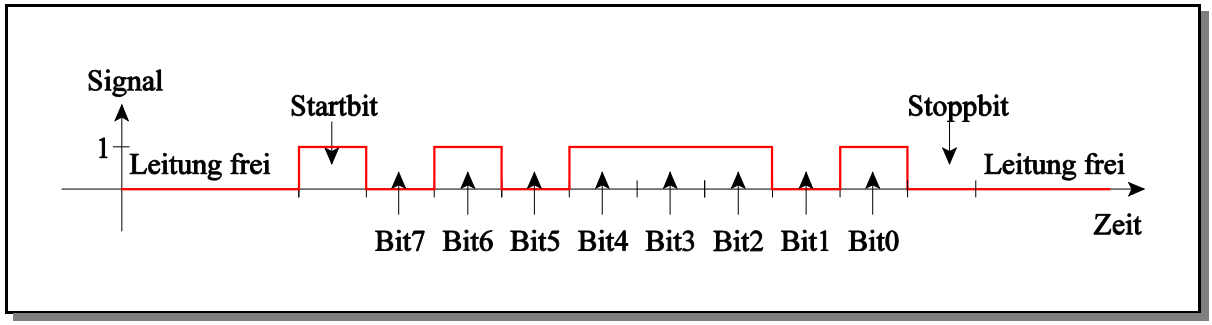

Abb. 4: Das Sendeprotokoll für die serielle Datenübertragung

Der Empfänger muss die seriell übertragenen Bits wieder zum richtigen Bitmuster zusammen fügen. Der Empfänger wartet zunächst, bis der Startimpuls mit dem Wert 1 kommt und beginnt dann mit dem Einlesen des darauf folgenden Bitmusters. Das Einlesen muss nun im gleichen Takt der Bitzeit geschehen wie beim Senden. Um die Werte der einzelnen Bits zuverlässig erkennen zu können, wird immer in der Mitte der zugehörigen Bitzeit nachgesehen, welchen Wert das gerade empfangene Datenbit hat. Will der Empfänger nach dem Startimpuls die Mitte des Bits 0 erreichen, muss er zunächst die 1,5 fache Bitzeit abwarten, bis er das Bit 7 liest. Danach wartet er eine einfache Bitzeit ab und liest dann das Datenbit 6 ein. Auf dieselbe Weise fährt er fort bis zum Datenbit 0. Den Abschluss bildet noch einmal eine Bitzeit, um in das Stoppbit zu kommen. Dieses Bit wird aber nicht mehr ausgewertet. Während des Abfragens ordnet der Empfänger die empfangenen Bits wieder zu einem korrekten Bitmuster an.

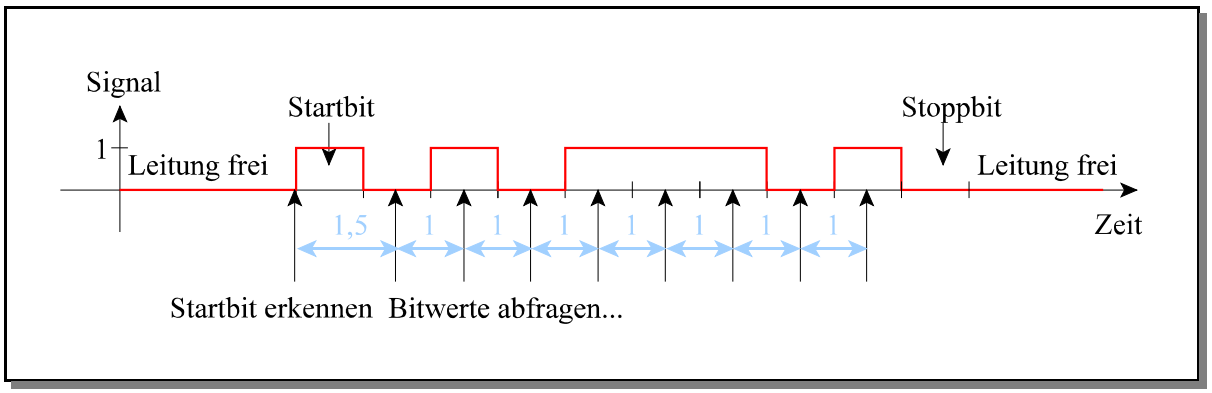

Abb. 5: Das Empfangsprotokoll für die serielle Datenübertragung

Abschließend wird das Bitmuster als Dualzahl gedeutet, die zugehörige Zahl im Zehnersystem gebildet und das entsprechende Zeichen ermittelt. Damit ist ein einziges Zeichen übertragen.

Datenfernübertragung mit der Platine

Zur Datenfernübertragung werden jetzt die beiden Platinen mit den erforderlichen Bauteilen versehen: Bei der Sender-Platine wird die Leuchtdiode an DTR und GND angeschlossen. Auf der Empfänger-Platine wird der Fototransistor zwischen dem Eingang DSR und dem auf High gelegten Ausgang DTR angeschlossen.

Abb. 6: Programm zur Datenfernübertragung

Das Sende-Dokument soll wie in Abb. 6 aussehen. Wir überlegen zunächst die Sende-Funktionen:

```
function init()
{
     COMX.open(1);
     COMX.DTR = 0;
}

function schluss()
{
     COMX.close();
}

function senden()
{
     var code;
     var bitmuster;
     var text = sform.text.value + "&"; // & als Ende-Zeichen
     var anzahl = text.length
     for (var i = 0; i<anzahl; i++)
     {
          code = text.charCodeAt(i);
          bitmuster = dec2dual(code);
          codeSenden(bitmuster);
     }
     alert("Übertragung beendet!");
}

function dec2dual(zahl)
{
     var stufenzahl = new Array(1,2,4,8,16,32,64,128);
     var s = ""; var rest = zahl;
     for (var i = 7; i>=0; i--)
     {
          bit = Math.floor(rest / stufenzahl[i]);
```

```
            rest = rest % stufenzahl[i];
            s = s + bit;
      }
      return s;
}

function codeSenden(b)
{
      //vgl. Aufgabe 7
}

function bitSenden(bit)
{
      var zeit = sform.zeit.value;
      COMX.DTR=bit;
      COMX.delay(zeit);
}
```

Es folgen die Funktionen für den Empfänger. Da er nicht die Länge des Textes kennt, beendet er seine Arbeit, wenn das Textende-Zeichen „&“ empfangen worden ist.

```
function init()
{
      COMX.open(1);
      COMX.DTR = 1;
}

function empfangen()
{
      vgl. Aufgabe 8
}

function empfangeBuchstabe()
{
      warteAufStartbit();
      var eineinhalbBitzeit = eform.zeit.value*1.5;
      COMX.delay(eineinhalbBitzeit);
      var code = empfangeByte();
      return String.fromCharCode(code);
}

function warteAufStartbit()
{
      while (COMX.DSR==0) { }
}

function empfangeByte()
{
      code = 0;
      var stufenzahl = new Array(1,2,4,8,16,32,64,128);
      for (var i = 7; i>=0; i--)
      {
```

```
            code = code + empfangeBit()*stufenzahl[i];
      }
      return code;
}

function empfangeBit()
{
      var bit = COMX.DSR;
      COMX.delay(eform.zeit.value);
      return bit;
}
```

Zur Übertragung des Textes muss zunächst das Empfänger-Programm gestartet werden, indem die Funktion `empfangen()` aufgerufen wird. Erst anschließend kann der Sendevorgang begonnen werden. Andernfalls würde das Empfänger-Programm das Startbit verpassen.

Aufgaben

1. Zeichne das vollständige Zeitdiagramm für die Übertragung des Buchstabens „B“.

2. Welches Zeichen wird in Abb. 4 übertragen?

3. Warum kann man nicht auf das Stoppbit verzichten?

4. Erläutere die Funktion `senden()`.

5. Welcher Programmierstil (Top-Down oder Bottom-Up) wurde beim Senden-Programm benutzt?

6. Erkundige dich nach der Bedeutung der `Math.floor`-Funktion und des Operators %. Welchen Wert gibt die Funktion `dec2dual` zurück, wenn sie als Parameter den Wert 79 erhält? Führe die Rechnungen der einzelnen Anweisungen des Funktionsrumpfes Schritt für Schritt durch, um den Rückgabewert zu bestimmen.

7. Schreibe die Funktion `codeSenden(b)`. Benutze dazu die `bitSenden`-Funktion.

8. Schreibe die Funktion `empfangen()`. Benutze die Funktion `empfangeBuchstaben()`. Achte darauf, dass die Funktion abbrechen soll, wenn das Ende-Zeichen & empfangen worden ist.

9. Erläutere die Funktion `empfangeBuchstabe()`.

10. Teste die beiden Programme aus. Du findest sie auf der CD im Verzeichnis `source\PDV`. Benutze dabei zunächst eine Bitzeit von 500 ms. Versuche daraufhin, die Bitzeit zu verringern. Bis zu welchen Bitzeiten arbeitet die Übertragung noch zuverlässig?