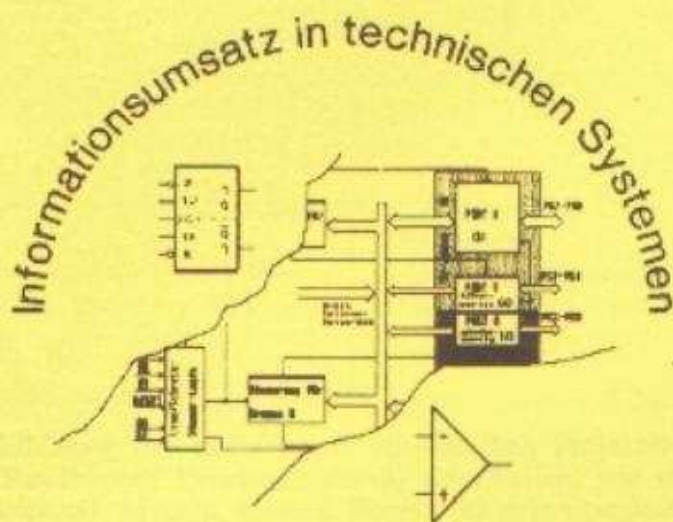


# TECHNIK



Messen, Steuern, Regeln  
mit dem Computer  
Ergänzung zur Centronics-Schnittstelle  
**Schnittstellenerweiterung**

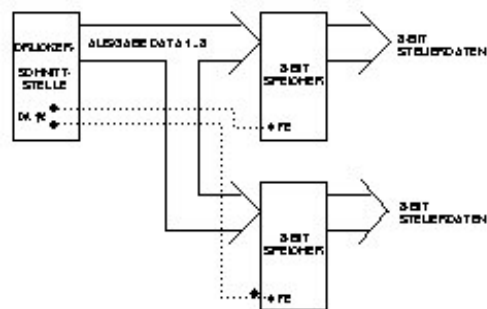
Peter Grimm  
Gesamtschule Castrop-Rauxel

2/92

Herausgeber:

Technik-Unterricht: Forum e.V. (TUF)  
Memelstraße 75 4100 Duisburg 1

# TECHNIK



## Schnittstellenerweiterung Ergänzung zur Centronics-Schnittstelle

# Inhalt

## 1. Vorüberlegungen (Problematisierung)

### Schaltplanübersicht

- 2.1 Schnittstellenerweiterung-1
- 2.2 Schnittstellenerweiterung-2
- 2.3 Schnittstellenerweiterung-3

### Software

- 3.1 Schnittstellenerweiterung-1
- 3.2 Schnittstellenerweiterung-3

### Platinenservice

- 4.1 8-BIT-Zwischenspeicher
- 4.2 Demultiplexer
- 4.3 8 UND-Gatter

Für die Durchführung des nachfolgend vorgestellten Projektes ist eine Centronics-Schnittstelle (Parallelport/ Druckeranschluß) erforderlich, wie sie im TUF-Heft 3/90 unter dem Arbeitstitel „Messen, Steuern, Regeln mit dem Computer“ erläutert wurde.

## Vorüberlegungen

Die vorliegenden Aufzeichnungen sind angeregt, entstanden durch ein Unterrichtsprojekt, das im Wahlpflichtbereich II (Jahrgangsstufe 9/10) durchgeführt wurde. Das Projekt kann aber auch unter dem Gesichtspunkt Optimierung technischer Systeme in den Jahrgangsstufen 11/1 bzw. 12/2 behandelt werden. Da die Hard- und Softwarelösungen auch für weitere, umfangreiche Steuerungsaufgaben einsetzbar sind, werden sie hier vorgestellt. Bei der Erweiterung des Problems Ampelsteuerung durch das Problem „Grüne Welle“ wird den Schülern sofort klar, daß mit der Centronics-Schnittstelle nur eine Kreuzung angesteuert werden kann. Bei einer „einfachen“ Kreuzung sind nur 6-Bits zur Steuerung der Lampen erforderlich, die gegenüberliegenden Ampeln können ja parallel angeschlossen werden. Eine zweite Kreuzung läßt sich ohne Hardwareerweiterung nicht mehr anschließen und steuern.

Um das Problem zu lösen, wollen wir jede Kreuzung mit einem separaten Speicher versehen, in den das Bitmuster zur Steuerung des Ampelablaufplans eingeschrieben wird. Da das Bitmuster aber an jedem Speicher (Kreuzung) anliegt (Bussystem), muß dafür gesorgt werden, daß nur der gewünschte Speicher die Information übernimmt. Dies wird dadurch gewährleistet, daß der gewünschte Speicher kurzfristig geöffnet (Freigabeingang) wird, dadurch wird die am Bus anliegende Information durchgeschleust, die Information liegt sofort am Ausgang an. Anschließend wird der Freigabeingang gesperrt, das eingeschriebene, durchgeschleuste Bitmuster wird im Speicher „eingefroren“. Durch gezieltes Freigeben des gewünschten Speichers kann jede beliebige Information in die Speicher (Kreuzung) geschrieben werden.

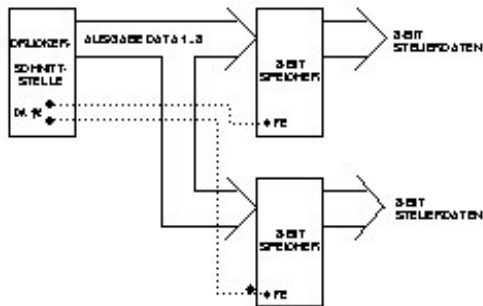
In dem vorliegenden Skript werden 3 mögliche Lösungen vorgestellt:

- 1. Schnittstellenerweiterung-1** (Ansprechen von 2 Speichern)  
Ausgabe des Speicherwertes über die Leitungen DA1..8  
Die Freigabeingänge FE werden über DATA1 und 2 direkt gesteuert
- 2. Schnittstellenerweiterung-2** (Ansprechen von 4 Speichern)  
Ausgabe des Speicherwertes über die Leitungen DA1..8  
Die Freigabeingänge FE werden über einen Multiplexer gesteuert, der über DATA1 und 2 angesprochen wird.  
(Die Multiplexer-Platine erlaubt auch das Ansprechen von 8 Speichern, da aber nur DATA1 und 2 für die Ansteuerung zur Verfügung stehen, lassen sich nur 4 Speicher steuern)
- 3. Schnittstellenerweiterung-3** (Ansprechen von 8 Speichern)  
Ausgabe des Speicherwertes über die Leitungen DA1..8  
Die Freigabeingänge FE werden über einen Adressenspeicher gesteuert.  
Der Adressenspeicher wird über DATA1, die eigentlichen Speicher über den Adressenspeicher und DATA2 freigegeben.

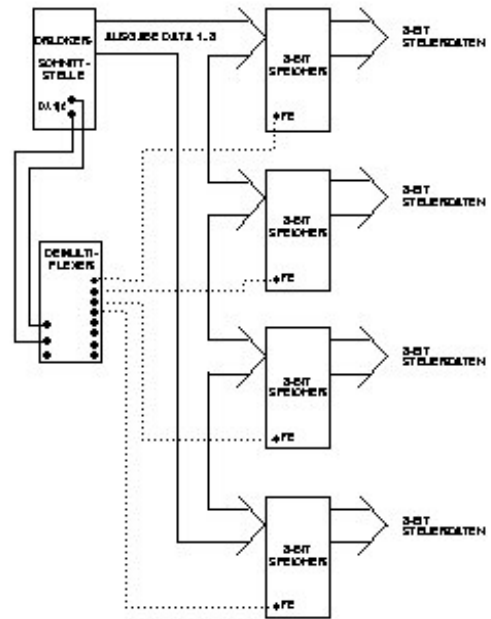
Hardware: Für alle 3 Lösungen kann der gleiche Zwischenspeicher verwendet werden. Für die zweite Lösung ist zusätzlich ein Multiplexer, für die dritte eine UND-Gatterplatine erforderlich, die nur dann ein 8-Bitmuster (hier die Speicheradresse) durchläßt, wenn die zweiten Eingänge der Und-Gatter über einen gemeinsamen Treibereingang (Transistorstufe, negiertes Verhalten) mit 0-Signal (Leitung DATA2) beaufschlagt werden.

Die Hardware wird komplett vorgestellt, bei der Software beschränke ich mich auf Hilfen zur Programmierung der Lösungen 1 und 3. Die Software ist wieder in Turbo-Pascal 3.0 geschrieben. Eine Begründung für diese Programmiersprache und Version sei an dieser Stelle kurz gegeben. Turbo-Pascal erlaubt strukturiertes Programmieren. Die Version 3.0 besitzt für den Technikunterricht einen ausreichenden Befehlsvorrat, die Version zeichnet sich durch eine überschaubare, kleine Benutzeroberfläche aus und benötigt nur einen Speicherplatz von ca. 40kbyte.

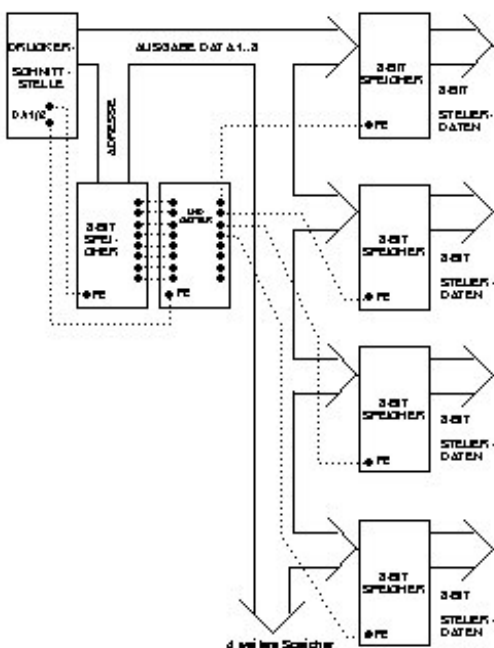
### Schnittstellenerweiterung-1



### Schnittstellenerweiterung-2



### Schnittstellenerweiterung-3



## Program Schnittstellenerweiterung\_1;

{  
Mit Hilfe der Speicherplatine (IC 74LS373) können 8-BIT Daten zwischengespeichert werden. Der Eingang QE der Platine wird fest mit GND verbunden. Die DATA-Ausgänge 1..8 (z.B. Port[888]) der Adapterplatine werden mit den Eingängen E1..8 und der Freigabe-eingang FE mit dem Ausgang DA1 (z.B. Port[890]) verbunden. Mit den Ausgängen DA1 und 2 lassen sich zwei Speicherplatinen direkt steuern. Das nachfolgende Programm zeigt in einfacher Form, wie die 8-BIT Daten in 2 Zwischenspeicher transportiert werden.  
}

```
const ioaus = 888;           {Drucker-Adresse}
var Wert : integer;
```

{Die nachfolgende Prozedur soll einen Eingabewert in einen adressierten Speicher schreiben.}

```
Procedure Wert_Uebernahme(Speicher:byte);
begin
  case speicher of
    1 : Port[ioaus+2]:=0;      {DA1 erhält H-Signal}
    2 : Port[ioaus+2]:=5;      {DA2 erhält H-Signal}
  end;
  Port[ioaus] :=Wert;         {Wert wird ausgegeben und
                              vom Speicher übernommen}
  Port[ioaus+2]:=1;          {Die 1 bewirkt, daß DA1 und 2 L-Signal
                              besitzen. Die Inhalte der angeschlosse-
                              nen Speicherplatinen ändern sich nicht.}

end;

begin
  clrscr;
  Wert:=0;                   {Schaffen einer definierten Anfangs-
                              bedingung, beide Speicher 0 setzen.}
  Wert_Uebernahme(1);        {Speicher 1 ansprechen}
  Wert_Uebernahme(2);        {Speicher 2 ansprechen}

  write ('Eingabe Speicherwert-1 : '); Readln(Wert);
  Wert_Uebernahme(1);

  write ('Eingabe Speicherwert-2 : '); Readln(Wert);
  Wert_Uebernahme(2);

end.
```

## Program Schnittstellenerweiterung\_3;

```
{  
Wenn eine Speicherplatine als Zwischenspeicher für die Adressen dient, können 8 Speicherplatinen  
angesteuert werden. Zusätzlich ist eine UND-Gatterschaltung zur Steuerung der Freigabeeingänge der  
Speicherplatinen erforderlich. Die Eingänge QE der Speicherplatinen und der Adresscodierplatine wer-  
den fest mit GND und die DATA-Ausgänge 1..8 (z.B. Port[888]) werden mit den Eingängen E1..8  
(parallel) verbunden. Der FE-Eingang der Adressspeicher-platine wird mit dem Ausgang DA1, der FE  
Ein-gang der UND-Gatterplatine mit DA2 verbunden. Die UND-Bedingung ist für alle 8 Gatter nur dann  
erfüllt, wenn die 2. Gatter-Eingänge über DA2 mit 1 belegt werden. Die Freigabeeingänge FE der 8  
Datenspeicherplatinen werden mit den Ausgängen 1..8 der UND-Gatteratine verbunden. Das nachfolgen-  
de Programm zeigt in einfacher Form, wie die 8-BIT Daten in 8 Zwischenspeicher transportiert werden.  
}
```

```
const ioaus = 888;           {Druckerport LPT1}  
var Nr,Wert : Byte;
```

{Die nachfolgende Prozedur soll einen Eingabewert in einen adressierten Speicher schreiben.}

```
Procedure Wert_Uebernahme(SpeicherNr,SpeicherWert:byte);  
var SP_Nr : byte;
```

```
begin
```

```
  writeln('Nr: ',SpeicherNr,' Wert : ',SpeicherWert);
```

```
  case SpeicherNr of
```

```
    1 : SP_Nr:=1;  2 : SP_Nr:=2;
```

```
    3 : SP_Nr:=4;  4 : SP_Nr:=8;
```

```
    5 : SP_Nr:=16; 6 : SP_Nr:=32;
```

```
    7 : SP_Nr:=64; 8 : SP_Nr:=128;
```

```
  end;
```

```
  Port[ioaus] := SP_Nr;
```

{der auf dem „Datenbus“ ausgegebene Wert soll nur in den Adresspeicher übernommen werden}

```
  Port[ioaus+2] := 4;
```

{DA1 erhält H-Signal, der Adresspeicher ist frei für eine Datenübernahme, DA2=1, d.h. der Transistor schaltet durch, alle UND-Gatter haben auf dem 2. Eingang 0-Signal, alle Speicherplatinen sind für eine Datenaufnahme gesperrt}

```
  Port[ioaus+2] := 5;
```

{Die 5 bewirkt, daß DA1 L-Signal und DA2 H-Signal besitzt. Die Inhalte der angeschlossenen Speicherplatinen (auch der Adressspeicherplatine) ändern sich nicht.

```
  Port[ioaus] := SpeicherWert;
```

{Wert auf „Datenbus“ ausgeben }

```
  Port[ioaus+2] := 1;
```

{Die 1 bewirkt, daß beide Ausgänge L-Signal erhalten. Die UND-Gatterplatine schaltet die oben angegebene Adresse durch (da der Transistor nicht durchschaltet, liegt H-Signal am 2. Gattereingang der UND-Gatter an). Der anliegende Speicherwert wird übernommen}

```
  Port[ioaus+2] := 5;
```

{alle Speicher gesperrt, der Speicherwert wird „eingefroren“}

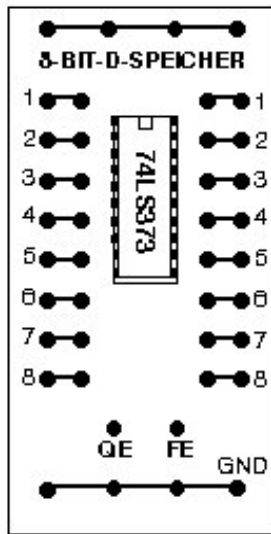
```
end;
```

```
Procedure Speicher_Null_setzen;  
var i:byte;  
begin  
  writeln('— Speicher_Null_setzen —');  
  for i:= 1 to 8 do Wert_Uebernahme(i,0);  
  writeln('— Ende Null-setzen —');  
  writeln;  
end;
```

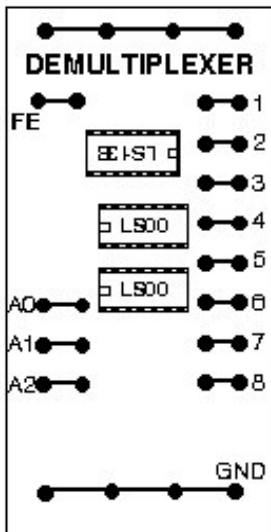
**{Demo - Hauptprogramm}**

```
begin  
  clrscr;  
  Speicher_Null_setzen;                                { Schaffen einer definierten Anfangs-  
                                                       bedingung, alle 8-Speicher 0 setzen. }  
  Wert_Uebernahme(1,5);                               { Speicher 1 ansprechen, 5 übergeben }  
  Wert_Uebernahme(2,255);                             { Speicher 2 ansprechen, 255 übergeben }  
  Wert_Uebernahme(8,8);                               { Speicher 8 ansprechen, 8 übergeben }  
  
  writeln;  
  
  write ('Eingabe Speicher_Nr ? : '); Readln(Nr);  
  write ('Eingabe Speicherwert : '); Readln(Wert);  
  
  Wert_Uebernahme(Nr,Wert);  
  writeln;  
  writeln('ENDE');  
end.
```

## 8-Bit-Zwischenspeicher



Eingänge			Ausgang
QE	FE	E1..8	Q1..8
L	H	H	H
L	H	L	L
L	L	X	keine Änderung
H	X	X	hochohmig



## 3-Bit-Binärdecoder/Demultiplexer

## 8 UND-Gatter mit gemeinsamen 2. Eingang

