4.7 PC-Hardware/Peripherie

4.7.1 E/A-Bausteine

E/A-Bausteine[1] ermöglichen den Anschluss analoger und digitaler Peripherie an einen Computer. Die meisten Computer haben mehrere solcher E/A-Bausteine an Bord.

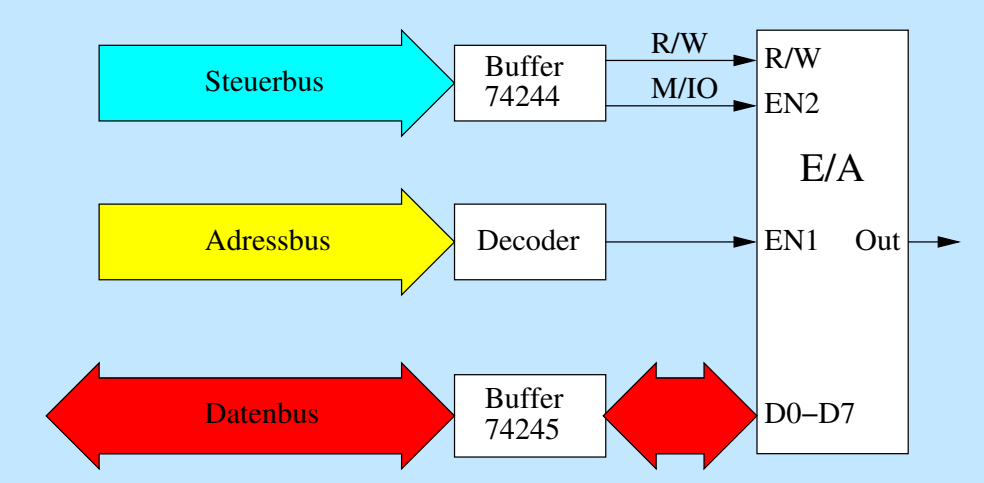

Abbildung 1: Anschluss eines E/A-Bausteinses an ein Bussystem

4.7.1.1 Schema und Signal-Zeit-Diagramm Die Abbildung 1 zeigt, wie man einen E/A-Baustein an ein Bussystem anschließen kann.

- Die Pufferbausteine 74244 (unidirektional) und 74245 (bidirektional) können dabei auch weggelassen werden.
- Das EN- (=Enable) oder CS- (=Chip Select) Signal wählt den Baustein aus. Falls wie hier mehrere solche Anschlüsse vorhanden sind, werden sie intern UND-verknüpft. Andernfalls muss man extern eine UND-Verknüpfung vornehmen.
- R/W (Read/Write) wählt die Schreib- oder Leserichtung aus.
- M/IO (Memory/IO) wählt zwischen Speicher- oder E/A-Bausteinen
- D0 – D7 (Data0 bis Data7) sind acht Leitungen des Datenbusses. Falls wie so oft am Bus mehr Leitungen vorhanden sind, werden diese überzähligen Leitungen nicht benutzt.

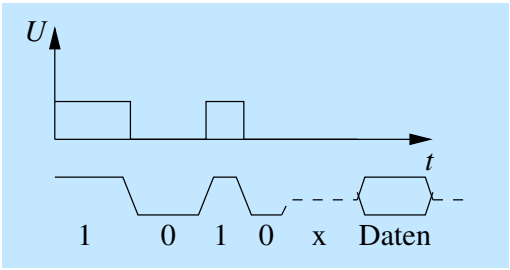

Abbildung 2: Beispiel eines Signal-Zeit-Diagramms

[1] E/A=Eingabe/Ausgabe, genauso I/O=Input/Output

Mit einem Signal-Zeit-Diagramm kann der zeitliche Ablauf auf einem Bussystem zum Beispiel beim Zugriff auf einen E/A-Baustein dargestellt werden. Abbildung 2 zeigt ein Beispiel eines Signal-Zeit-Diagramms. In Abbildung 3 sieht man das Signal-Zeit-Diagramm eines Schreibzugriffs

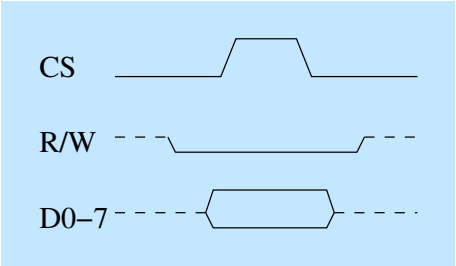

Abbildung 3: Signal-Zeit-Diagramm des Zugriffs auf den E/A-Baustein

auf den E/A-Baustein.

4.7.1.2 Auswahl: Speicher oder E/A Es sind zwei verschiedene Wege üblich, wie man zwischen Speicher- und E/A-Zugriff unterscheiden kann:

a) **Separate I/O** – Es gibt eine Leitung (M/IO), mit der unterschieden wird zwischen Speicher und E/A. Bei traditionellen PC-Prozessoren ist diese Leitung vorhanden. Je nach Prozessor-Befehl wird diese Leitung aktiviert. Somit sind die Prozessor-Befehle aufgespalten in normale und in E/A-Befehle. Vorteil ist, dass der gesamte Adressbereich einmal für Speicher und dann noch einmal für E/A zur Verfügung steht. Nachteil ist, dass der Befehlssatz des Prozessors durch die Extra-Befehle unnötig aufgebläht wird. Außerdem gibt es für E/A bei diesen Prozessoren nur einige spezielle Befehle, so dass die E/A-Programmierung umständlich ist.

b) **Memory Mapped I/O** – Hier gibt es keine Leitung M/IO. Bei vielen Motorola-Prozessoren ist das so der Fall. Je nach Adresse kann entweder ein Speicher- oder ein E/A-Baustein gemeint sein. Der Befehlssatz muss nicht zwischen Speicher- und E/A-Befehlen unterscheiden. Alle Befehle, die auf Speicherworte (Datenworte im Speicher) anwendbar sind, können auch für E/A-Bausteine genutzt werden. Der Nachteil liegt darin, dass ein Teil des Adressraums für E/A-Bausteine verloren geht und damit nicht mehr für RAM genutzt werden kann. Der klassische PC gibt ein Beispiel ungünstiger Aufteilung:

   - `0000.0000 - 0009.FFFF` : 640 kiB RAM
   - `000A.0000 - 000B.FFFF` : 128 kiB Video-RAM (E/A!)
   - `000F.0000 - 000F.FFFF` : max. 64 kiB ROM
   - `0010.0000 - FFFF.FFFF` : 4095 MiB RAM

4.7.1.3 Auswahl des einzelnen Bausteins Wie wird nun der Adressdecoder aus Abbildung 1 realisiert? Auch hier gibt es wieder verschiedene Möglichkeiten. Abbildung 4 zeigt die dual codierte Adressauswahl. Hier z.B. wird der Baustein ausgewählt, falls die Adresse gleich `11 1011 1101` ist. Vorteilhaft ist die große Anzahl möglicher Bausteine (bei 10 Leitungen sind es 1024). Nachteilig ist der hohe Aufwand. Abbildung 5 zeigt die linear codierte Adressauswahl. In diesem Beispiel wird der Baustein ausgewählt bei der Adresse `00 0000 1000`; der Aufwand ist sehr niedrig, die Decodierung sehr schnell. Nachteilig ist aber die sehr geringe Anzahl an möglichen Bausteinen (hier 10). Bei Auftreten mehrerer Einsen in der Adresse (z.B. bei Adresse 11.1111.1111) werden eventuell mehrere Bausteine gleichzeitig angesprochen; es ist dann möglich, dass Fehlfunktionen oder Zerstörung auftreten.

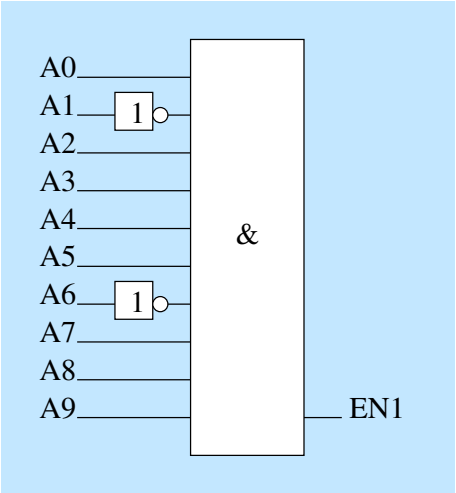

Abbildung 4: Dual codierte Adressauswahl

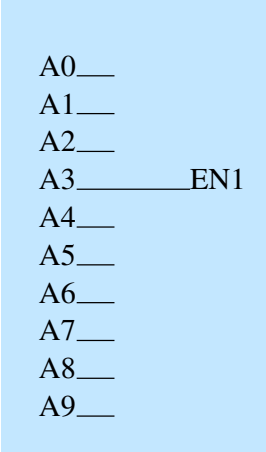

Abbildung 5: Linear codierte Adressauswahl

4.7.2 Schnittstellen-Typen

Während die Computer-Seite der E/A-Bausteine oft nach dem gleichen Schema aufgebaut ist, unterscheidet sich die andere, die Schnittstellen-Seite der E/A-Bausteine stark. Man kann z.B. folgende Einteilung vornehmen:

- Schnittstellen für analoge Signale
- Schnittstellen für digitale Signale
  - parallel (Raum-Multiplex)
    - unidirektional
    - bidirektional
  - seriell (Zeit-Multiplex)
    - synchron (mit Takt)
    - asynchron (ohne Takt)

4.7.3 Beispiel: Serielle Schnittstellen

Ein noch immer wichtiges Beispiel für eine serielle Schnittstelle ist die in RS232 und V.24 genormte serielle Schnittstelle. Sie ermöglicht mit nur zwei Leitungen (und Masse) eine asynchrone, bidirektionale vollduplex-Verbindung. Mit einer Leitung (plus Masse) ist eine asynchrone, unidirektionale Verbindung möglich. Abbildung 6 zeigt ein Beispiel für die Übertragung eines Bytes über eine RS232-Verbindung. Begonnen wird immer mit dem Start-Bit, es folgen die (hier)

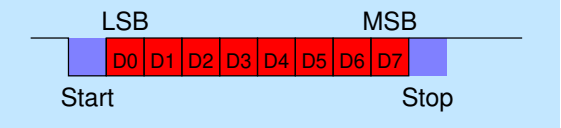

Abbildung 6: Übertragung eines Bytes mit 19200,1N8

acht Datenbits (beginnend mit dem *least significant bit* D0) und ein Stopbit. Jedes Bit hat hier die Dauer $t = \frac{1}{19200}\text{s} = 52{,}1\,\mu\text{s}$. Es gibt hier kein Paritäts-Bit. Diese Einstellung kann auf der Wind.-Eingabeaufforderung mit dem Befehl `mode com1: 19200,1N8` (*19200 baud, 1 stop bit, no parity, 8 data bits*) vorgenommen werden. Realisiert wird die RS232-Schnittstelle meistens mit einem UART-Baustein (UART=*universal asynchronous receiver and transmitter*).

Nach dem OSI-Modell kann man die RS232-Schnittstelle wie so beschreiben:

a) Schicht 0 (mechanisch/elektrisch):

- 9-, 15- oder 25-poliger Sub-D-Stecker
- unsymmetrische Übertragung; Low-Potential $-3\,\text{V} \ldots -15\,\text{V}$, High-Potential $3\,\text{V} \ldots 15\,\text{V}$, bei Datenbits ist Low=1, High=0, bei Steuerbits ist Low=0, High=1

b) Schicht 1 (Bitübertragung):

- Signal-Zeit-Diagramm siehe Abbildung 6
- Hardware-Flusskontrolle per RTS- und CTS-Leitung möglich
- Software-Flusskontrolle per XON/XOFF-Protokoll möglich (Strg - S =stop, Strg - Q =weiter)

c) Schicht 2 (Link/Sicherung): Zwischen dem letzten Daten- und dem ersten Stopbit kann ein *Parity*-Bit übertragen werden, mit dem die Zahl der 1-Bits pro Wort auf eine gerade (*even*) oder eine ungerade (*odd*) Anzahl ergänzt wird.

d) Schicht 3 (Netzwerk/Routing): entfällt, da es sich um eine Punkt-zu-Punkt-Verbindung handelt. Bei TCP/IP kann diese Funktionalität durch die Protokolle SLIP und PPP übernommen werden.

e) Schicht 4 – 7: wird durch Anwender-Software übernommen, im einfachsten Fall durch ein Terminalprogramm (z.B. Minicom).

f) zu Schicht 6 (Zeichensatz): bei Übertragung alphanumerischer Daten kann (je nach Anzahl der Datenbits) CCITT Nr. 2 (5 Bit), ASCII (7 Bit) oder jeder andere Zeichensatz (8 Bit) verwendet werden.