# 1. Einführung

Das vorliegende Buch betrifft das Problem, den PC wie einen Mikrocontroller einzusetzen. Hierbei betrachten wir drei Auslegungen (Abbildung 1.1, Tabelle 1.1):

1. Der PC arbeitet selbst als Mikrocontroller, verhält sich also – erweitert mit entsprechenden Interfaceschaltungen – wie ein PIC, ein AVR, ein 8051 oder ein x-beliebiger anderer Mikrocontroller. Im Gegensatz zu den Simulatoren der gängigen Entwicklungsumgebungen wollen wir aber echte Peripherie mit der Geschwindigkeit echter Mikrocontroller-Schaltkreise betreiben.
2. Der PC arbeitet als Steuerrechner. Die zu steuernde Hardware ist über allgemein übliche Interfaces angeschlossen. Die jeweilige Anwendungsaufgabe wird durch PC-Programmierung gelöst, wobei die Anwendungssoftware typischerweise unter Steuerung eines marktgängigen Betriebssystems läuft.
3. Der PC arbeitet mit außen angeschlossenen Mikrocontrollern zusammen. Die zeitkritischen (und womöglich knifflig zu programmierenden) Steuerungsaufgaben werden außen erledigt (vom Mikrocontroller), die Aufgaben der Programmentwicklung, Bedienung, Anzeige, Datenverwaltung, Netzwerkkommunikation usw. hingegen innen (vom PC).

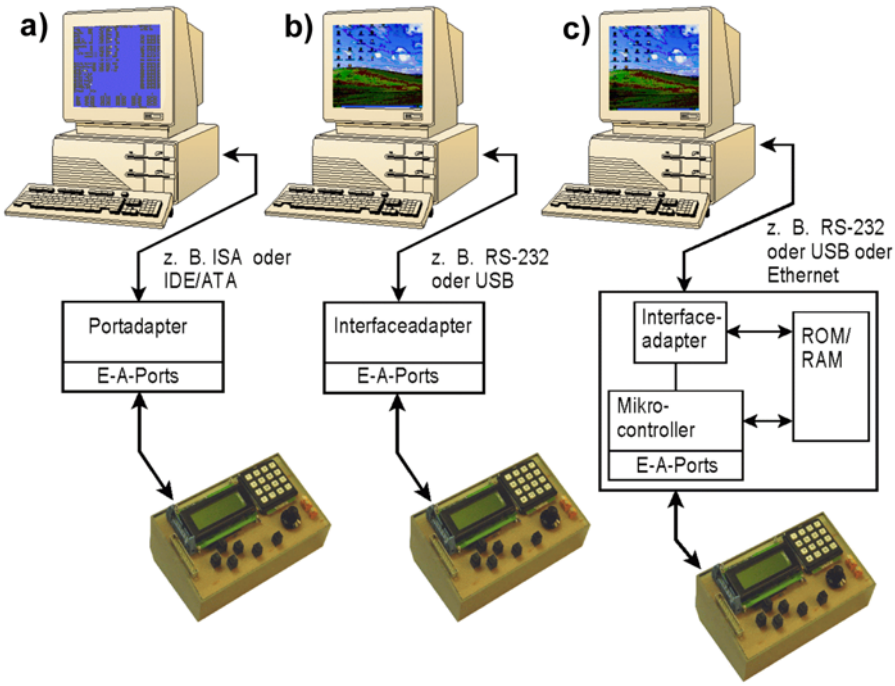


***Abb.1.1*** PCs im Einsatz. Drei Auslegungen: a) als Mikrocontroller, b) als Steuerrechner, c) mit angeschlossenem Mikrocontroller. Es ist eine x-beliebige Peripherie zu steuern (im Beispiel eine Bedientafel mit LCD-Anzeige, Tastenmatrix, Einzeltasten und Incrementalgeber)

| PC-Konfiguration | als Mikrocontroller | als Steuerrechner | mit angeschlossenem Mikrocontroller |
|---|---|---|---|
| Reaktionszeiten an den anwendungsseitigen Schnittstellen | Mikrosekunden oder noch kürzer | Millisekunden | Mikrosekunden oder noch kürzer |
| Programmierung (PC-Software) | hardware-nah | übliche Anwendungs-programmierung | übliche Anwendungs-programmierung |
| Hardware-Plattform | beliebig | moderne PCs gemäß Anforderungen des Be-triebssystems | moderne PCs gemäß Anforderungen des Be-triebssystems |
| Betriebssystem | keines, DOS, Linux, EFI | Windows, Linux | Windows, Linux |
| typische PC-Schnittstel-len | ISA, IDE/ATA, PCI (und Nachfolger) | serielle (RS-232) oder parallele Schnittstelle, USB, Ethernet | serielle (RS-232) oder parallele Schnittstelle, USB, Ethernet |
| wird der PC für Anwendungsfunktio-nen ausgenutzt?*) | möglich | ja | möglich |

*): betrifft Bedienung, Anzeige, Dateiverwaltung, Netzwerkkommunikation usw.

***Tabelle 1.1*** Einzelheiten zu den Konfigurationen von Abbildung 1.1

Das ist an sich nichts Neues. Seit es Personalcomputer (PCs) gibt, hat man sie nicht nur als Büromaschinen und Spielkonsolen verwendet, sondern auch zum Entwickeln von Programmen für Mikrocontroller und zum Steuern x-beliebiger Einrichtungen.

Moderne Personalcomputer setzen derartigen Anwendungen jedoch einigen Widerstand entgegen:

- komplizierte Systemsoftware verhindert den direkten Zugriff auf die Hardware,
- komplizierte neumodische Interfaces erschweren das Entwickeln anwendungsspezifischer Lösungen (Hardware + steuernde Software).

Es ist also erforderlich, sich etwas einfallen zu lassen. Unser Ehrgeiz beschränkt sich hierbei auf vergleichsweise bescheidene Vorhaben. Die Bescheidenheit betrifft sowohl den Umfang der Hardware als auch die zum Programmieren erforderlichen Aufwendungen. Damit entfällt die wirklich harte Windows- oder Linux-Programmierung (vor allem: die Entwicklung eigener Gerätetreiber). Zudem scheiden alle Entwicklungen aus, die zur Mitgliedschaft in Standardisierungsgremien und zur Zahlung von Lizenzgebühren zwingen.

**Typische Anwendungsgebiete:**

- Einstieg in die Mikrocontroller-Programmierung,
- Selbstbauvorhaben,
- Plattformen zum Experimentieren und Lernen, die über die typischen Starterkits, Simulatoren usw. hinausgehen,

- Erproben, Bewerten und Auswählen,
- Entwicklungsvorhaben (die Anwendungslösung wird auf dem PC entwickelt und anschließend auf den echten Mikrocontroller portiert),
- komplette Systemlösungen, z. B. in der Meß- und Prüftechnik.

**Welche PCs?**

- ältere Modelle (Weiternutzen, Aufbrauchen),
- moderne PCs der Massenfertigung (SHV[1)]),
- moderne PC-Baugruppen, vor allem kleine Motherboards, PC/104-Moduln, Single Board Computer (SBCs) usw.

**Der PC als Mikrocontroller**

Wir verwenden einen PC anstelle eines Mikrocontrollers. Um die typischen Mikrocontroller-Ports nachzubilden, wird der PC mit Zusatzschaltungen erweitert. Eine solche Anordnung mag in geradezu groteskem Maße unwirtschaftlich erscheinen (Abbildung 1.2), hat aber – in bestimmten Anwendungsbereichen – Vorteile, die auf der Hand liegen:

- sie ist – aufs Ganze gesehen – preiswerter als ein gängiges Starterkit, das schließlich auch noch einen PC braucht (Entwicklungsumgebung),
- sie ist in verschiedener Hinsicht leistungsfähiger,
- sie ist viel komfortabler (Bedienung, Anzeige, Fehlersuchunterstützung[2)], Dateiverwaltung),
- die Turnaround-Zeiten (vom Programmieren bis zum lauffähigen Programm) sind viel kürzer,
- es ist kein Download[3)] erforderlich; das Programm ist sofort nach dem Compilieren lauffähig,
- wir können preisgünstige oder gar kostenlose Software einsetzen (es sind „nur“ PCs zu programmieren)[4)],
- wir haben nahezu ungeschränkte Freiheiten zum Ändern und Experimentieren – es ist möglich, bekannte Mikrocontroller-Architekturen zu verbessern und eigene Vorstellungen zur Systemarchitektur zu verwirklichen.

---

1) SHV = Standard High Volume, also all das, was beim nächsten PC-Händler ohne weiteres beschafft werden kann.

2) Wenn man es richtig anstellt, kann die im PC laufende Software (der Emulator) nie abstürzen, ganz gleich, wie fehlerhaft das eigentliche Anwendungsprogramm ist – somit haben wir die Chance, auch die verzwicktesten Abläufe bis in die Einzelheiten hinein zu verfolgen.

3) Das Programmieren eines Bytes in einem Flash-ROM dauert typischerweise 1...5 ms. Rechnen wir mit durchschnittlich 2 ms. 1 kBytes erfordern 2 s, 10 kBytes 20 s usw.

4) Vgl. demgegenüber die Preise mancher Programmentwicklungssysteme für Mikrocontroller – und zwar der Vollversionen, nicht der eingeschränkten Versionen, wie sie oft den Starterkits beiliegen.

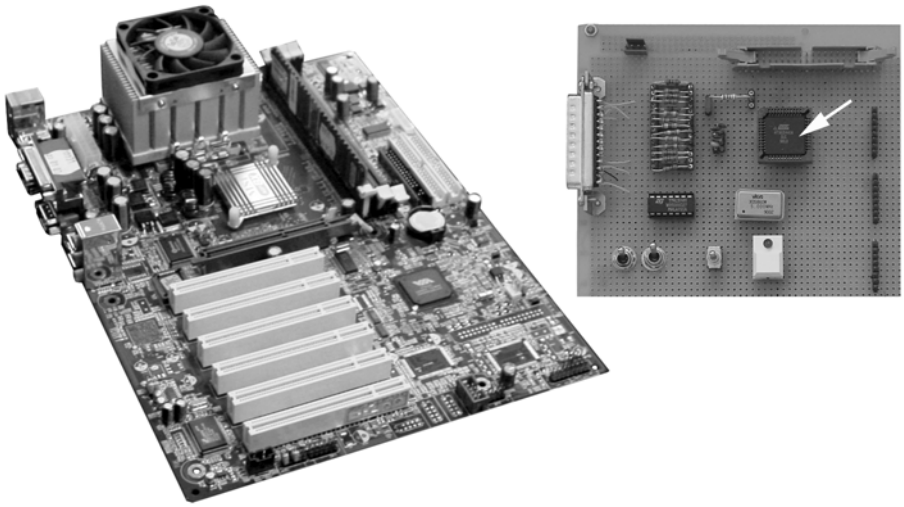

***Abb.1.2*** Es kann sein, daß der dicke Prozessor auf dem Motherboard nicht mehr leistet als der bescheidene Mikrocontroller (Pfeil)

Für diesen Einsatzfall ist (1) auf Geschwindigkeit und (2) hardware-nah zu programmieren – es muß möglich sein, auf schnellstem Wege auf die Schnittstellen-Hardware zuzugreifen. Soll sich der Programmieraufwand in vertretbarem Rahmen halten (Umfang, Schwierigkeitsgrad), scheidet Windows als Systemplattform aus.

**Der PC mit angeschlossener Steuerungshardware**
Eine gleichsam klassische Nutzungsweise des PCs. Es geht darum, Leuchtdioden und Relais anzusteuern, Kontakte abzufragen usw. Hierbei wollen wir die Möglichkeiten des PCs in vollem Umfang ausnutzen. Beispiel: die Bedienung der angeschlossenen Einrichtungen über eine graphische Oberfläche (Abbildung 1.3). Hinsichtlich des Interfaces kommt es nicht auf Geschwindigkeit an. Es darf vergleichsweise langsam sein (Millisekunden). Die ergänzende Hardware darf aber nicht allzu viel kosten – und sie muß sich ohne besonderen Aufwand (eigens zu schreibende Gerätetreiber usw.) von üblichen Entwicklungsumgebungen aus ansprechen lassen.

**Der PC mit angeschlossenem Mikrocontroller**
Wenn wir alles zugleich haben wollen – kürzeste Reaktionszeiten an den Schnittstellen plus Komfort eines modernen Betriebssystems (sprich: Windows) plus annehmbaren Aufwand –, so können wir uns nur auf jene Programmschnittstellen und Interfaces stützen, die vom System sozusagen freiwillig unterstützt werden. Dann müssen wir aber die Steuerung schneller E-A-Abläufe aus dem PC auslagern. Der PC ist dann nur noch Entwicklungs- und Bedienplattform[1)].

**Soll der PC für die Anwendungsfunktionen ausgenutzt werden?**
Stellen wir uns eine x-beliebige Anwendung vor, z. B. die Steuerung eines Parkscheinautomaten oder eines Meßgerätes. Es gibt zwei Möglichkeiten der Auslegung, die beide ihre Anwendungsgebiete haben:

1) der Unterschied zum herkömmlichen Starterkit: PC und Mikrocontroller sollen nicht nur zu Entwicklungszwecken miteinander verbunden werden, sondern auch im Anwendungsfall eine Einheit bilden.

- der PC leistet nur das, was ansonsten der Mikrocontroller tun würde. Mikrocontroller haben keine Bildschirme und Festplatten. Zur Bedienung unseres Gerätes ist ein entsprechendes Bedienfeld anzuschließen, zum Speichern irgendwelcher Daten könnte ein Flash-ROM vorgesehen werden usw. Wenn alles läuft (nach Abschluß der Entwicklung) braucht der PC weder Bildschirm noch Tastatur. Das ist die herkömmliche Nutzungsweise im Rahmen von kleineren Embedded Systems.
- die übliche Ausstattung des PCs wird für die Anwendungsfunktionen ausgenutzt. Der Bildschirm dient als Anzeigetafel und – im Zusammenwirken mit Maus und Tastatur – als Bedienfeld, die Festplatte dient zum Speichern von Daten, die Netzwerkschnittstelle zur Fernbedienung usw. Die offensichtlichen Vorteile:
  - es ist weniger Hardware selbst zu entwickeln,
  - die Ausstattung der typischen PCs ist kostengünstig (SHV),
  - wir müssen nicht so viel selbst programmieren, da für viele Funktionen die gängige System- und Anwendungssoftware ausgenutzt werden kann.

  Diese Auslegung hat in den letzten Jahren mehr und mehr Bedeutung erlangt. Das betrifft vor allem Anwendungsfälle mit hohem Bedienkomfort, komplizierten Algorithmen und großem Datenumfang. Der entscheidende Gesichtspunkt: die Nutzung der allgemein verfügbaren Standard-Software (Betriebssysteme, Anwendungspakete, Compiler usw.).

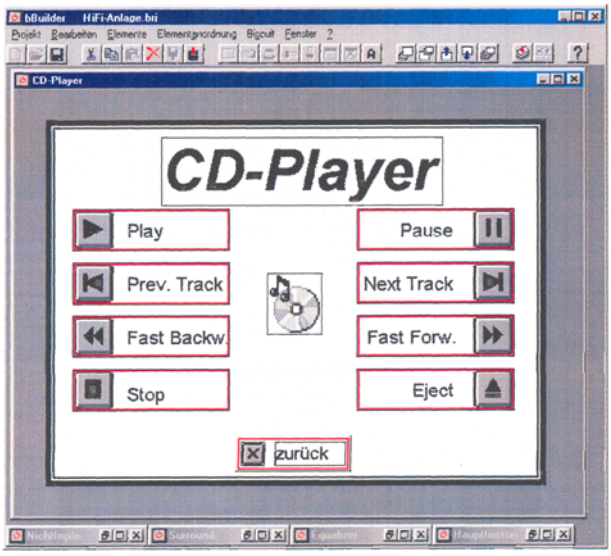


***Abb.1.3*** Bedienung eines CD-Spielers über Windows – ein Demonstrationsbeispiel