Dr. Helmut Bähring

Modul 63711

Anwendungsorientierte Mikroprozessoren

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Das Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere das Recht der Vervielfältigung und Verbreitung sowie der Übersetzung und des Nachdrucks bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder ein anderes Verfahren) ohne schriftliche Genehmigung der FernUniversität reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhalt von KE1:

Diese Seite bleibt aus technischen Gründen frei

1. Einführung

1.1 Definitionen und Begriffe

Der **Mikroprozessor**[1] (µP, *Micro Processing Unit* – MPU) ist die Zentraleinheit (*Central Processing Unit* – CPU) eines Digitalrechners, die auf einem einzigen Halbleiterplättchen (*Chip*) untergebracht ist. Er umfaßt die beiden Komponenten Steuerwerk und Operationswerk (bzw. Rechenwerk), die untereinander Informationen über Steuer- und Meldesignale austauschen (s. Bild 1.1-1). Während das Steuerwerk für den zeitgerechten Ablauf der Befehlsbearbeitung eines Programms zuständig ist, führt das Operationswerk (auch Rechenwerk genannt) die eigentlichen arithmetisch/logischen Operationen aus.

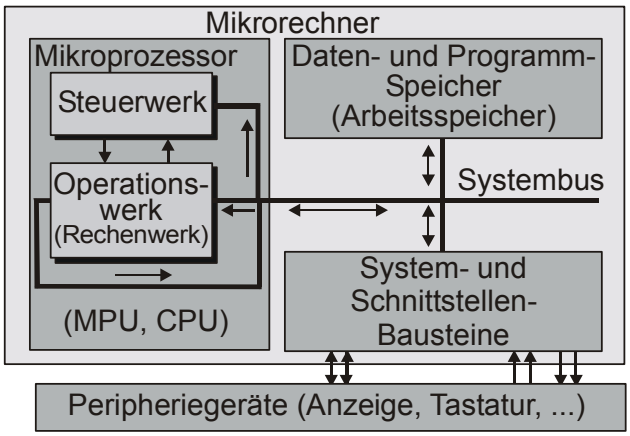

Bild 1.1-1: Aufbau eines Mikrorechners

Ein **Mikrorechner** (µR, Mikrocomputer) enthält neben dem Mikroprozessor als Zentraleinheit zusätzlich noch einen Daten- und Programmspeicher, die wir zusammenfassend als Arbeitsspeicher bezeichnen. Die Verbindung des Speichers mit dem Mikroprozessor geschieht über den (bidirektionalen) Systembus[2]. Er leitet die Befehle aus dem Speicher zum Steuerwerk des Prozessors, die Operanden vom Speicher zum Operationswerk und die Ergebnisse zurück zum Speicher. Daten und Programme können gemischt in einem einzigen Speicher untergebracht oder aber in getrennten Daten- bzw. Programm-Speichern abgelegt sein. Zum Mikrorechner gehören weiterhin Schnittstellen-Bausteine für den Anschluß diverser Peripheriegeräte sowie weitere Systembausteine[3] (Zähler, Echtzeituhr, Digital/Analog- bzw. Analog/Digital-Wandler usw., vgl. Kapitel 9).

Unter einem **Mikrorechner-System** (µRS, Mikrocomputer-System – MCS) verstehen wir einen Mikrorechner mit allen angeschlossenen Peripheriegeräten, der sog. **Peripherie** des Systems. Dazu gehören z.B. Festplatten, Tastatur und Bildschirm, Drucker, aber auch einfache Sensoren, Aktoren, Anzeigen usw.

---

[1] Die Vorsilbe „Mikro-“ bezieht sich nur auf die Größe des Prozessors, nicht jedoch auf seine Leistungsfähigkeit oder die Art der Realisierung des Steuerwerks. Die Namensgebung geschah zu einer Zeit, als der Prozessor eines Digitalrechners noch ein Schrank großes Gehäuse füllte.

[2] Unter einem **Bus** versteht man eine Sammlung von mehreren Leitungen, auf denen gleichartige Informationen parallel und in dieselbe Richtung übertragen werden und an denen alle Komponenten derart angeschlossen sind, daß zu jedem Zeitpunkt höchstens eine senden, aber alle anderen empfangen können.

[3] Wir sprechen in diesem Kurs vereinfachend auch dann von „Bausteinen“, wenn es sich dabei um Komponenten handelt, die auf demselben Chip untergebracht sind.

Nach den Grundzügen ihrer Architektur unterscheiden wir bei den Prozessoren zwischen den folgenden Typen:

- **CISC-Prozessoren** (*Complex Instruction Set Computer*) verfügen über einen (relativ) umfangreichen Befehlssatz. Sie sind meist mikroprogrammiert, d.h. sie besitzen ein Mikroprogramm-Steuerwerk (MPStW), dessen Mikroprogramme vom Hersteller in einem Festwertspeicher (*Control Memory/Store*) untergebracht wurden und vom Benutzer nicht geändert werden können. Ihre Architektur folgt (meist) dem von-Neumann-Prinzip, d.h. Befehle und Daten liegen im selben Arbeitsspeicher gemischt vor und werden über ein einziges Bussystem[4] transportiert (vgl. Bild 1.1-1). Diese Prozessoren werden heute hauptsächlich in Steuerungssystemen verwendet. In früheren Jahren dominierten die CISC-Prozessoren der Intel-80x86-Familie (bis 80486) den Markt der Personal Computer (PC).
- **RISC-Prozessoren** (*Reduced Instruction Set Computer*) sind nicht mikroprogrammierte Mikroprozessoren. Sie besitzen statt dessen ein fest verdrahtetes Steuerwerk, bei dem die Befehle durch direkte (kombinatorische) Schaltlogik realisiert werden. Die Bezeichnung RISC stammt von dem (ursprünglich) relativ kleinen Befehlssatz dieser Prozessoren. RISC-Prozessoren besitzen üblicherweise eine sog. Harvard-Architektur, bei der Befehle und Daten in getrennten Speichern untergebracht und gleichzeitig über getrennte Bussysteme transportiert werden können (s. Bild 1.1-2). Sie werden hauptsächlich in den Mikrorechnern der höheren Leistungsklasse für anspruchsvolle Steuerungsaufgaben eingesetzt.

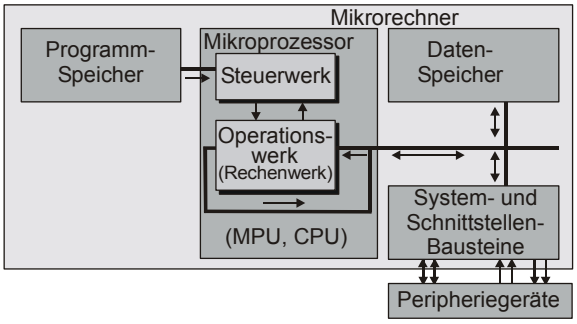

Bild 1.1-2: Aufbau eines Mikrorechners mit Harvard-Architektur

- **Hybride CISC/RISC-Prozessoren**, stellen eine Mischform mit CISC- und RISC-Eigenschaften dar. Die Mehrzahl der heute im universellen Rechnerbereich eingesetzten Hochleistungs-Mikroprozessoren gehört dieser Klasse an. Insbesondere in den Personal Computern sind sie millionenfach vertreten (als Intel Pentium oder dazu ‚kompatible' Prozessoren).

Nach der Art ihres Einsatzes unterscheiden wir die folgenden µP-Klassen:

- **Universelle Mikroprozessoren** (*General Purpose Processors*) sind Prozessoren ohne spezielle Schnittstellen oder Komponenten, wie sie z.B. in PCs oder Laptops eingesetzt werden. Mit geeigneten Programmen und externen Erweiterungsbausteinen sind sie für alle allgemeinen Anwendungen einsetzbar. Universelle Mikroprozessoren können sowohl vom Typ CISC, RISC sein oder eine Mischform aus beiden sein.

[4] Da zu jedem Taktzyklus nur jeweils ein Befehl oder ein Datum transportiert werden kann, bildet dieses Bussystem den sog. von-Neumann-Flaschenhals (*Bottleneck*).

- **Anwendungsorientierte Mikroprozessoren** sind Mikroprozessoren für spezielle Anwendungen. Dazu gehören:
  - die **Mikrocontroller** (µC, *Microcontroller Unit* – MCU), die über besondere Schnittstellen, integrierte Speicher und festgelegte Programme zur Steuerung bestimmter Vorgänge verfügen. Sie stellen – nach unserer Klassifikation – vollständige Mikrocomputer auf einem einzigen Chip dar (*Computer on a Chip, Single-Chip Computer*). Im Unterschied zu den Prozeßrechnern der Vergangenheit, die – oft schrankgroß – ‚von außen' ein System steuerten, können Mikrocontroller wegen ihrer geringen Größe direkt in dieses System integriert werden (s. Abschnitt 1.2). Sie werden nach dem CISC- oder RISC-Prinzip gebaut.
  - die **Digitalen Signalprozessoren** (*Digital Signal Processor* – DSP), die besonders zur digitalen Verarbeitung analoger Signale mit speziellen Befehlen und Hardwarekomponenten konzipiert sind. Ihr Rechenwerk ist auf die schnelle Reihenberechnung (*Multiply-Accumulate* – MAC) ausgelegt, die in vielen Algorithmen der digitalen Signalverarbeitung eine zentrale Rolle spielt.
  - Immer häufiger verfügen Mikrocontroller aber auch über erweiterte Rechenwerke und Speicher-/Bus-Architekturen, die eine effiziente Verarbeitung analoger Signale ermöglichen. Andererseits besitzen DSPs häufig aber auch eine große Anzahl von integrierten Schnittstellen zur Kommunikation und Steuerung. Sie werden dann oft als **Digitale Signalcontroller** (DSC) bezeichnet.

1.2 Einsatzgebiete anwendungsorientierter Mikroprozessoren (Th. Ungerer)

Das wesentliche Einsatzgebiet anwendungsorientierter Mikroprozessoren, insbesondere der Mikrocontroller, sind die sogenannten **eingebetteten Systeme** (*Embedded Systems*). Hierunter versteht man Datenverarbeitungssysteme, die in ein technisches Umfeld integriert sind. Dort stellen sie ihre Datenverarbeitungsleistung zur Steuerung und Überwachung dieses Umfeldes zur Verfügung. Ein Beispiel für ein eingebettetes System ist etwa die mit einem Mikrocontroller oder Mikroprozessor realisierte Steuerung einer Kaffeemaschine. Hier dient die Datenverarbeitungsleistung dazu, die umgebenden Komponenten wie Wasserbehälter, Heizelemente und Ventile zu koordinieren, um einen guten Kaffee zu bereiten. Der PC auf dem Schreibtisch zu Hause ist hingegen zunächst kein eingebettetes System. Er wirkt dort als reines Rechnersystem zur Datenverarbeitung für den Menschen. Ein PC kann jedoch ebenfalls zu einem eingebetteten System werden, sobald er z.B. in einer Fabrik zur Steuerung einer Automatisierungsanlage eingesetzt wird.

Eingebettete Systeme sind weit zahlreicher zu finden als reine Rechnersysteme. Ihr Einsatzgebiet reicht von einfachen Steuerungsaufgaben für Haushaltsgeräte, Unterhaltungselektronik oder Kommunikationstechnik über Anwendungen in der Medizin und in Kraftfahrzeugen bis hin zur Koordination komplexer Automatisierungssysteme in Fabriken. In unserem täglichen Leben sind wir zunehmend von solchen Systemen umgeben. Das folgende Bild 1.2-1 zeigt über der Zeitachse eine Auswahl von Geräten, durch die man von morgens, 06:00 Uhr, bis abends, 22:00 Uhr, mit eingebetteten Systemen in Kontakt kommt.

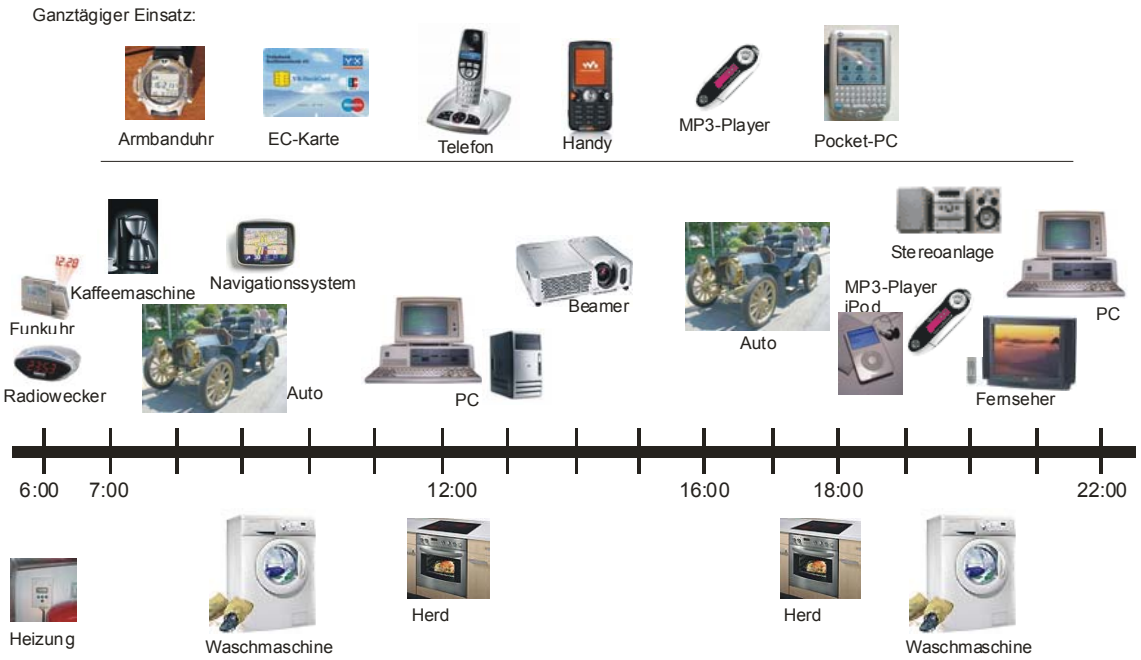

Bild 1.2-1: Eingebettete Systeme in Geräten des täglichen Lebens

Als einfaches Beispiel aus dieser Geräte-Palette wollen wir im Bild 1.2-2 den Aufbau eines sog. *MP3-Players* (MP3-Abspielgerät[1]) darstellen. Die zentrale Komponente des Gerätes ist ein großer, nicht flüchtiger Speicherbaustein (Flash-EEPROM), d.h. er verliert auch nach dem Ausschalten der Betriebsspannung nicht seinen Inhalt.

Die Umwandlung der gespeicherten digitalen MP3-Dateien wird vom integrierten Digitalen Signalprozessor (DSP) und dem Digital/Analog-Wandler (DAC) vorgenommen. Der nachgeschaltete Verstärker (*Amplifier* – AMP) sorgt für die vom Kopfhörer benötigte elektrische Spannung und Leistung. Gesteuert werden der Speicherbaustein, der DSP und die anderen Peripheriekomponenten wie Tasten und Flüssigkristall-Anzeige (*Liquid Crystal Display* – LCD) vom Mikrocontroller (µC), der über eine USB-Schnittstelle (*Universal Serial Bus*) und den Anschlußstecker mit einem PC verbunden werden kann. Über den USB-Anschluß und eine integrierte Ladeschaltung kann bei manchen Geräten auch der Akku geladen werden.

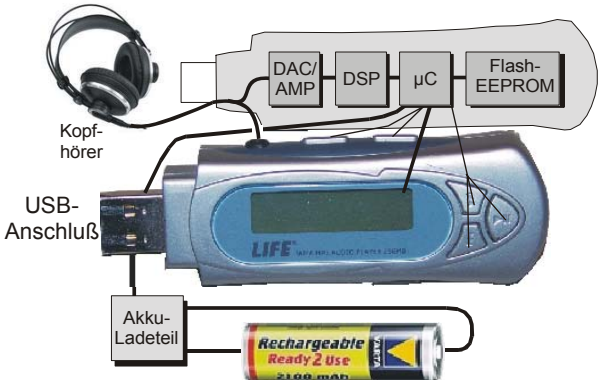

Bild 1.2-2: Aufbau eines MP3-Players

[1] MP3 steht für MPEG-1 Audio Layer 3, MPEG für Moving Picture Experts Group

Gegenüber reinen Rechnersystemen werden an eingebettete Systeme einige zusätzliche Anforderungen gestellt:

- **Schnittstellenanforderungen:** Umfang und Vielfalt von Ein-/Ausgabeschnittstellen ist bei eingebetteten Systemen üblicherweise höher als bei reinen Rechnersystemen. Dies ergibt sich aus der Aufgabe, eine Umgebung zu steuern und zu überwachen. Hierzu müssen die verschiedensten Sensoren und Aktuatoren (Aktoren) bedient werden, welche über unterschiedliche Schnittstellentypen mit dem eingebetteten System verbunden sind.
- **Mechanische Anforderungen:** An eingebettete Systeme werden oft erhöhte mechanische Anforderungen gestellt. Für den Einsatz in Fabrikhallen, Fahrzeugen oder anderen rauhen Umgebungen muß das System robust sein und zahlreichen mechanischen Belastungen standhalten. Aus diesem Grund werden z.B. gerne spezielle, mechanisch stabile Industrie-PCs eingesetzt, wenn – wie oben erwähnt – ein PC als eingebettetes System dient. Normale PCs würden den mechanischen Anforderungen einer Fabrikhalle oder eines Fahrzeugs nicht lange standhalten. Weiterhin werden der zur Verfügung stehende Raum und die geometrische Form für ein eingebettetes System in vielen Fällen vom Umfeld diktiert, wenn dieses System in das Gehäuse eines kleinen Gerätes wie z.B. eines Telefons untergebracht werden muß.
- **Elektrische Anforderungen:** Die elektrischen Anforderungen an eingebettete Systeme beziehen sich meist auf den Energieverbrauch und die Versorgungsspannung. Diese können aus verschiedenen Gründen begrenzt sein. Wird das System in eine vorhandene Umgebung integriert, so muß es aus der dortigen Energieversorgung mitgespeist werden. Versorgungsspannung und maximaler Strom sind somit vorgegebene Größen. In mobilen Systemen werden diese Größen von den Eigenschaften einer Batterie oder eines Akkumulators diktiert. Um eine möglichst lange Betriebszeit zu erzielen, sollte hier der Energieverbrauch so niedrig wie möglich sein. Ein niedriger Energiebedarf ist auch in Systemen wichtig, bei denen die Abwärme gering gehalten werden muß. Dies kann erforderlich sein, wenn spezielle isolierende Gehäuse (wasser- oder gasdicht, explosionsgeschützt usw.) den Abtransport der Abwärme erschweren oder die Umgebungstemperatur sehr hoch ist.
- **Zuverlässigkeitsanforderungen:** Bestimmte Einsatzgebiete stellen hohe Anforderungen an die Zuverlässigkeit eines eingebetteten Systems. Fällt z.B. die Bremsanlage eines Kraftfahrzeugs oder die Steuerung eines Kernreaktors aus, können die Folgen katastrophal sein. In diesen Bereichen muß durch spezielle Maßnahmen sichergestellt werden, daß das eingebettete System so zuverlässig wie möglich arbeitet und für das verbleibende Restrisiko des Ausfalls ein sicherer Notbetrieb möglich ist.
- **Zeitanforderungen:** Eingebettete Systeme müssen oft in der Lage sein, bestimmte Tätigkeiten innerhalb einer vorgegebenen Zeit auszuführen. Solche Systeme nennt man **Echtzeitsysteme**. Reagiert z.B. ein automatisch gesteuertes Fahrzeug nicht rechtzeitig auf ein Hindernis, so ist eine Kollision unvermeidlich. Erkennt es eine Abzweigung zu spät, wird es einen falschen Weg einschlagen.
- Eine weitere Anforderung an Echtzeitsysteme ist die längerfristige **Verfügbarkeit**. Dies bedeutet, daß ein solches System seine Leistung über einen langen Zeitraum hinweg unterbrechungsfrei erbringen muß. Betriebspausen, z.B. zur Reorganisation interner Datenstrukturen, sind nicht zulässig.

Der Echtzeit-Aspekt bedarf einiger zusätzlicher Erläuterungen. Allgemein betrachtet, unterscheidet sich ein Echtzeitsystem von einem Nicht-Echtzeitsystem dadurch, daß zur logischen Korrektheit die zeitliche Korrektheit hinzukommt. Das Ergebnis eines Nicht-Echtzeitsystems ist korrekt, wenn es den logischen Anforderungen genügt, d.h. z.B. ein richtiges Resultat einer Berechnung liefert. Das Ergebnis eines Echtzeitsystems ist nur dann korrekt, wenn es logisch korrekt ist und zusätzlich zur rechten Zeit zur Verfügung steht. Anhand der Zeitbedingungen können drei **Klassen von Echtzeitsystemen** unterschieden werden:

- **Harte Echtzeitsysteme** sind dadurch gekennzeichnet, daß die Zeitbedingungen unter allen Umständen eingehalten werden müssen. Man spricht auch von harten Zeitschranken. Das Überschreiten einer Zeitschranke hat katastrophale Folgen und kann nicht toleriert werden. Ein Beispiel für ein hartes Echtzeitsystem ist die bereits oben angesprochene Kollisionserkennung bei einem automatisch gesteuerten Fahrzeug. Eine zu späte Reaktion auf ein Hindernis führt zu einem Unfall.
- **Feste Echtzeitsysteme** definieren sich durch Zeitschranken, sog. feste Zeitschranken, deren Überschreitung das Ergebnis einer Berechnung zwar wertlos macht, ohne daß die Folgen unmittelbar katastrophal sind. Ein Ergebnis, das nach der Zeitschranke geliefert wird, kann verworfen werden, und es wird auf das nächste Ergebnis gewartet. Ein Beispiel ist die Positionserkennung eines automatischen Fahrzeugs. Eine zu spät gelieferte Position ist wertlos, da sich das Fahrzeug mittlerweile weiterbewegt hat. Dies kann zu einer falschen Fahrstrecke führen, die jedoch gegebenenfalls später korrigiert werden kann. Allgemein sind feste Echtzeitsysteme oft durch Ergebnisse oder Werte mit Verfallsdatum gekennzeichnet.
- **Weiche Echtzeitsysteme** besitzen Zeitschranken, die eher eine Richtlinie als eine harte Grenze darstellen. Man spricht deshalb auch von weichen Zeitschranken. Ein Überschreiten um einen gewissen Wert kann toleriert werden. Ein Beispiel ist die Beobachtung eines Temperatursensors in regelmäßigen Abständen für eine Temperaturanzeige. Wird die Anzeige einmal etwas später aktualisiert, so ist dies häufig nicht weiter schlimm.

Die zeitliche Vorhersagbarkeit des Verhaltens spielt für ein Echtzeitsystem die dominierende Rolle. Eine hohe Verarbeitungsgeschwindigkeit ist eine „nette Beigabe“, jedoch bedeutungslos, wenn sie nicht genau bestimmbar und vorhersagbar ist. Benötigt beispielsweise eine Berechnung in den meisten Fällen nur eine Zehntelsekunde, jedoch in wenigen Ausnahmefällen eine ganze Sekunde, so ist für ein Echtzeitsystem nur der größere Wert ausschlaggebend. Wichtig ist immer der schlimmste Fall der Ausführungszeit (*Worst Case Execution Time* – WCET). Aus diesem Grund sind z.B. moderne Mikroprozessoren für Echtzeitsysteme nicht unproblematisch, da ihr Zeitverhalten durch ihre leistungssteigernden Techniken (auf die hier nicht näher eingegangen werden kann) schwer vorhersagbar ist. Einfache Mikrocontroller ohne diese Techniken besitzen zwar eine weitaus geringere Verarbeitungsleistung, ihr Zeitverhalten ist jedoch genauer zu bestimmen.

Der Begriff ***Ubiquitous Computing*** wurde bereits Anfang der 90er Jahre des letzten Jahrhunderts[2] von der Firma Xerox geprägt und bezeichnet eine Zukunftsvision: Mit Mikroelektronik angereicherte Gegenstände sollen so alltäglich werden, daß die enthaltenen Rechner als solche nicht mehr wahrgenommen werden. Die Übersetzung von „*ubiquitous*“ ist „allgegenwärtig“ oder „ubiquitär“, synonym dazu wird oft der Begriff „*pervasive*“ im Sinne von „durchdringend“

[2] Wir lassen diesen Zusatz im weiteren Verlauf des Kurses weg, da dies zu keinen Mißverständnissen führen kann.

benutzt.

Ubiquitäre Systeme sind eine Erweiterung der eingebetteten Systeme. Als ubiquitäre (allgegenwärtige) Systeme bezeichnet man eingebettete Rechnersysteme, die selbstständig auf ihre Umwelt reagieren. Bei einem ubiquitären System kommt zusätzlich zu einem eingebetteten System noch Umgebungswissen hinzu, das es diesem System erlaubt, sich in hohem Maße auf den Menschen einzustellen. Die Benutzer sollen nicht in eine virtuelle Welt gezogen werden, sondern die gewohnte Umgebung soll mit Computerleistung angereichert werden, so daß neue Dienste entstehen, die den Menschen entlasten und ihn von Routineaufgaben befreien.

Betrachten wir das Beispiel eines Fahrkartenautomaten: Während bis vor einigen Jahren rein mechanische Geräte nur Münzen annehmen konnten, diese gewogen, geprüft und die Summe mechanisch berechnet haben, so ist der Stand der Technik durch eingebettete Rechnersysteme charakterisiert. Heutige Fahrkartenautomaten lassen eine Vielzahl von Einstellungen zu und arbeiten mit recht guter computergesteuerter Geldscheinprüfung. Leider muß der häufig überforderte Benutzer die Anleitung studieren und aus einer Vielzahl möglicher Eingabemöglichkeiten auswählen. In der Vision des *Ubiquitous Computing* würde beim Herantreten an den Fahrkartenautomaten der in der Tasche getragene „Persönliche Digitale Assistent“ (PDA) oder das Mobiltelefon („*Handy*“) über eine drahtlose Netzverbindung mit dem Fahrkartenautomaten Funkkontakt aufnehmen und diesem unter Zuhilfenahme des im PDA oder Handy gespeicherten Terminkalenders mitteilen, wohin die Reise voraussichtlich gehen soll. Der Fahrkartenautomat würde dann sofort unter Nennung des voraussichtlichen Fahrtziels eine Verbindung und eine Fahrkarte mit Preis vorschlagen, und der Benutzer könnte wählen, ob per Bargeldeinwurf gezahlt oder der Fahrpreis von der Geldkarte oder dem Bankkonto abgebucht werden soll.

In diesem Sinne wird der Rechner in einer dienenden und nicht in einer beherrschenden Rolle gesehen. Man soll nicht mehr gezwungen sein, sich mit der Bedienung des Geräts, sei es ein Fahrkartenautomat oder ein heutiger PC, auseinandersetzen zu müssen. Stattdessen soll sich das Gerät auf den Menschen einstellen, d.h. mit ihm in möglichst natürlicher Weise kommunizieren, Routinetätigkeiten automatisiert durchführen und ihm lästige Tätigkeiten, soweit machbar, abnehmen. Daraus ergeben sich grundlegende Änderungen in der Beziehung zwischen Mensch und Maschine. *Ubiquitous Computing* wird deshalb als die zukünftige dritte Phase der Computernutzung gesehen:

- Phase I war die Zeit der Großrechner, in der wegen seines hohen Preises ein Rechner von vielen Menschen benutzt wurde.
- Phase II, die heute vorliegt, ist durch die Mikrorechner geprägt. Diese sind preiswert genug, daß sich sehr viele Menschen einen eigenen Rechner leisten können – zumindest in den gut entwickelten Industriestaaten. Auch das Betriebssystem eines Mikrorechners ist üblicherweise für einen einzelnen Benutzer konfiguriert. Technische Geräte arbeiten heute meist schon mit integrierten Rechnern. Diese können jedoch nicht miteinander kommunizieren und sind nicht dafür ausgelegt, mittels Sensoren Umgebungswissen zu sammeln und für ihre Aktionen zu nutzen.
- Phase III der Computernutzung wird durch eine weitere Miniaturisierung und einen weiteren Preisverfall mikroelektronischer Bausteine ermöglicht. Diese Phase der ubiquitären Systeme soll es ermöglichen, den Menschen mit einer Vielzahl nicht sichtbarer, in Alltagsgegenstände

eingebetteter Computer zu umgeben, die drahtlos miteinander kommunizieren und sich auf den Menschen einstellen.

Fünf Merkmale sind zur Kennzeichnung **ubiquitärer Systeme** hervorzuheben:

- Sie sind eine Erweiterung der „eingebetteten Systeme“, also von technischen Systemen, in die Computer eingebettet sind.
- Sie integrieren eingebettete Computer überall und in hoher Zahl (Allgegenwart).
- Sie nutzen drahtlose Vernetzung; dazu zählen die Technologien der Mobiltelefone, Funk-LAN (*Local Area Network*), Bluetooth und Infrarot.
- Sie verwenden Umgebungswissen, das es ihnen erlaubt, sich in hohem Maße auf den Menschen einzustellen.
- Sie binden neue Geräte ein wie z.B. mobile „PCs“ (PDA, *Handheld*), Mobiltelefone und am Körper getragene („*wearable*“) Rechner.

Technisch gesehen sind für ein ubiquitäres System viele kleine, oftmals tragbare, in Geräten oder sogar am und im menschlichen Körper versteckte Mikroprozessoren und Mikrocontroller notwendig, die über Sensoren mit der Umwelt verbunden sind und bisweilen auch über Aktuatoren (Aktoren) aktiv in diese eingreifen. Verbunden sind diese Rechner untereinander und mit dem Internet über drahtgebundene oder drahtlose Netzwerke, die oftmals im Falle von tragbaren Rechnern spontan Netzwerke bilden. Die Rechnerinfrastruktur besteht aus einer Vielzahl unterschiedlicher Hardware und Software: kleine tragbare Endgeräte, leistungsfähige Server im Hintergrund und eine Kommunikationsinfrastruktur, die überall und zu jeder Zeit eine Verbindung mit dem „Netz“ erlaubt. Ein wesentliches Problem für die Endgeräte stellt die elektrische Leistungsaufnahme und damit die eingeschränkte Nutzdauer dar. Ein weiteres Problem ist die Bereitstellung geeigneter Benutzerschnittstellen: Die Benutzer erwarten auf ihre Bedürfnisse zugeschnittene und einfach zu bedienende, aber leistungsfähige Dienste.

Die Einbeziehung von Informationen aus der natürlichen Umgebung der Geräte stellt ein wesentliches Kennzeichen ubiquitärer Systeme dar. Die Berücksichtigung der Umgebung, des „Kontexts“, geschieht über die Erfassung, Interpretation, Speicherung und Verbindung von Sensordaten. Oftmals kommen Systeme zur orts- und richtungsabhängigen Informationsinterpretation auf mobilen Geräten hinzu. Verfügt ein Gerät über die Information, wo es sich gerade befindet, so kann es bestimmte Informationen in Abhängigkeit vom jeweiligen Aufenthaltsort auswählen und anzeigen. Das Gerät paßt sich in seinem Verhalten der jeweiligen Umgebung an und wird damit ortssensitiv. Beispiele für ortssensitive Geräte sind die GPS-gesteuerten Kfz-Navigationssysteme (*Global Positioning System*), die je nach Ort und Bewegungsrichtung angepaßte Fahrhinweise geben. Die GPS-Infrastruktur stellt an jeder Position die geographischen Koordinaten zur Verfügung, und das Navigationssystem errechnet daraus die jeweilige Fahranweisung.

Eine Mobilfunkortung ermöglicht lokalisierte Informationsdienste, die je nach Mobilfunkregion einem Autofahrer Verkehrsmeldungen nur für die aktuelle Region geben oder nahe gelegene Hotels und Restaurants anzeigen. Viele Arten von Informationen lassen sich in Abhängigkeit von Zeit und Ort gezielt filtern und sehr stark einschränken. Der tragbare Rechner entwickelt so ein regelrecht intelligentes oder kooperatives Verhalten.

1.3 Energiespartechniken

Die Reduktion des Energiebedarfs stellt ein weiteres wichtiges Kriterium beim Rechnerentwurf dar – insbesondere von eingebetteten Systemen. Durch den Einsatz in immer kleiner werdenden mobilen Geräten ist die verfügbare Energiemenge durch Batterien oder Akkumulatoren begrenzt. Es gilt, möglichst lange mit der vorhandenen Energie auszukommen. Im wesentlichen lassen sich drei Ansätze unterscheiden:

- Verringerung der Taktfrequenz,
- Verringerung der Versorgungsspannung,
- Optimierung der Mikroarchitektur.

Die Verringerung der Taktfrequenz ist zunächst ein einfacher und wirkungsvoller Weg, den Energiebedarf zu reduzieren. Bei modernen CMOS-Schaltungen (*Complementary Metal-Oxide Semiconductor*) ist der Energiebedarf E proportional zur Taktfrequenz F, d.h.

$$E \sim F$$

Eine Halbierung der Taktfrequenz reduziert daher auch den Energiebedarf auf die Hälfte. Leider wird hierdurch die Verarbeitungsgeschwindigkeit ebenfalls halbiert.

Eine zweite Möglichkeit zur Reduktion des Energiebedarfs ist die Reduktion der Versorgungsspannung. Der Energieverbrauch ist proportional zum Quadrat der Spannung U, d.h.

$$E \sim U^2$$

Eine Reduktion der Spannung auf siebzig Prozent führt somit ebenso (ungefähr) zu einer Halbierung des Energiebedarfs.

Variiert man Versorgungsspannung und Taktfrequenz gleichzeitig, so erhält man:

$$E \sim F \cdot U^2$$

Versorgungsspannung und Taktfrequenz sind hierbei aber keine unabhängigen Größen. Je geringer die Versorgungsspannung, desto geringer auch die maximal mögliche Taktfrequenz. Näherungsweise kann hierfür ein linearer Zusammenhang angenommen werden:

$$F_{max} \sim U$$

Setzt man diesen Zusammenhang in die obige Gleichung für den Energiebedarf ein, so erhält man die **Kubus-Regel** für eine simultane Änderung der Taktfrequenz und Versorgungsspannung:

$$E \sim F^3 \quad \text{oder} \quad E \sim U^3$$

Eine Reduktion von Taktfrequenz und Versorgungsspannung verringert neben dem Energiebedarf auch die Verarbeitungsleistung. Forschungsansätze gehen deshalb dahin, Taktfrequenz und Versorgungsspannung eines Mikrocontrollers im Hinblick auf die Anwendung zu optimieren. Dazu wird z.B. der Versuch unternommen, in einem Echtzeitsystem Versorgungsspannung und Taktfrequenz eines Prozessors dynamisch an die einzuhaltenden Zeitschranken anzupassen. Besteht für eine Aufgabe sehr viel Zeit, so muß der Prozessor nicht mit voller Geschwindigkeit arbeiten, sondern kann mit reduzierter Taktfrequenz und damit reduziertem Energieverbrauch operieren.

Auch im Bereich der universellen Mikroprozessoren werden ähnliche Ansätze verfolgt. So regelte beispielsweise der Crusoe-Prozessor von Transmeta Taktfrequenz und Versorgungs-

spannung anhand der durchschnittlichen Prozessorauslastung. Diese wurde durch Messung der Prozessorruhezeiten (*Idle Times*) bei der Prozeßausführung ermittelt.

Weitere Ansätze versuchen, die Versorgungsspannung einer CMOS-Schaltung – abhängig von Betriebsparametern, wie etwa der Temperatur – derart zu regeln, daß die Schaltung gerade noch fehlerfrei funktioniert. Ein anderer Forschungszweig befaßt sich mit der Frage, die Mikroarchitektur eines Mikroprozessors derart zu optimieren, daß der Energiebedarf ohne Einbußen in der Verarbeitungsgeschwindigkeit gesenkt werden kann. Hier ist eine Reihe von Maßnahmen denkbar:

- **Reduktion der Busaktivitäten:** Bei konsequentem Einsatz der von RISC-Prozessoren bekannten ***Load-Store*-Architektur**, bei der nur durch die Lade- bzw. Speicherbefehle – und nicht z.B. durch einen arithmetischen Befehl – auf den Speicher zugegriffen wird, arbeiten die meisten Prozessor-Operationen auf den internen Registern. Die Busschnittstelle wird während dieser Zeit also nicht benötigt und kann als Energieverbraucher abgeschaltet bleiben. Ein entsprechend mächtiger interner Registersatz hilft ebenfalls, externe Buszugriffe zu verringern. Eine weitere Möglichkeit besteht darin, wo es sinnvoll ist, schmale Datentypen (8 oder 16 bit anstelle von 32 bit) zu verwenden. Dabei muß zur Datenübertragung nicht die volle Busbreite aktiviert werden.
- **Statisches Power-Management:** Dem Anwender können Befehle zur Verfügung gestellt werden, die Teile des Mikroprozessors abschalten (z.B. Speicherbereiche, Teile des Rechenwerks) oder den ganzen Mikrorechner in eine Art „Schlafmodus“ versetzen (vgl. Unterabschnitt 2.3.2). Dadurch kann der Anwender gezielt den Energieverbrauch kontrollieren.
- **Dynamisches Power-Management:** Neben dem statischen Power-Management kann der Prozessor dynamisch für jeden gerade bearbeiteten Befehl die zur Verarbeitung nicht benötigten Komponenten abschalten. Dies kann z.B. in den verschiedenen Stufen der Befehlsbearbeitung (s. Abschnitt 5.2, KE2) erfolgen.
- **Erhöhung der Code-Dichte:** Die Code-Dichte kennzeichnet einerseits die Anzahl der Befehle, die für eine Anwendung benötigt werden – was auch als „Mächtigkeit“ eines Befehlssatzes bezeichnet wird. Andererseits bezeichnet sie aber auch die durchschnittliche Anzahl von Bits, die für die Realisierung der Befehle benötigt werden.
  Je höher die Code-Dichte, desto weniger bzw. kürzere Befehle sind erforderlich. Eine hohe Code-Dichte spart Energie aus zwei Gründen: Erstens wird weniger Speicher benötigt und zweitens fallen weniger Buszyklen zur Ausführung einer Anwendung an. Von diesem Gesichtspunkt aus stellt eine RISC-Architektur also eher einen Nachteil dar, da durch die einfacheren Instruktionen konstanter Bitbreite die Code-Dichte gegenüber CISC-Architekturen im allgemeinen geringer ist.

Man sieht, daß durchaus komplexe Zusammenhänge zwischen Architektur und Energiebedarf bestehen. Es ist somit wünschenswert, möglichst frühzeitig bei der Entwicklung eines neuen Mikrorechners Aussagen über den Energiebedarf machen zu können. Neue Forschungsarbeiten zielen deshalb darauf ab, Modelle zur Vorhersage des Energiebedarfs zu entwickeln, die Aussagen in frühen Entwicklungsschritten erlauben. Ziel ist es, zusammen mit den ersten funktionalen Simulationen eines Mikroprozessors auch Daten über den wahrscheinlichen Energiebedarf zu erhalten.

2. Mikrocontroller

2.1 Einleitung

In diesem Kapitel wollen wir uns nun ausführlich mit den **Mikrocontrollern** (μC) beschäftigen. Wie bereits gesagt, können Mikrocontroller wegen ihrer geringen Größe direkt in das zu steuernde System integriert werden (s. Bild 2.1-1). Diese Controller nennt man deshalb auch ‚eingebetteten Controller' (*Embedded Controller*). Werden solche Mikrocontroller um weitere externe Komponenten, z.B. Speicher, Eingabe- und Anzeigemodule usw., zu einem Mikrorechner-System ergänzt, spricht man dementsprechend auch von „eingebetteten Systemen" (*Embedded Systems*). Die von ihnen gesteuerten technischen Geräte bezeichnet man als „eingebettete Anwendungen"[1] (*Embedded* (*Systems*) *Applications*). Bei komplexen Systemen sind häufig mehrere der im Bild dargestellten Komponenten ebenfalls Mikrocontroller (oder DSPs).

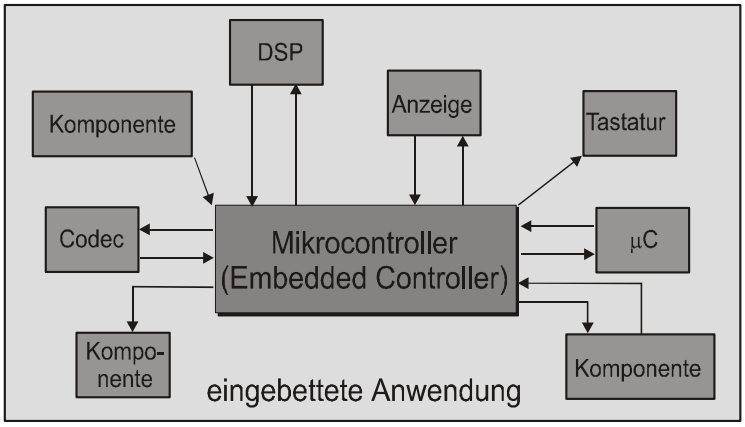

Bild 2.1-1: Einbettung eines Mikrocontrollers ins System

Im Hochleistungsbereich und in sehr komplexen Systemen wurden und werden aber auch heute oft noch Standard-Mikroprozessoren (mit den benötigten Peripheriemodulen) zur Steuerung und Regelung eingesetzt. So gab es Prozessorfamilien, die vollständig oder überwiegend in eingebetteten Anwendungen eingesetzt wurden, z.B. der RISC-Prozessor Am29000 der Firma AMD oder die 680X0-Prozessoren von Motorola. Nach ihrer Einsatzweise werden diese Mikroprozessoren als „eingebettete Mikroprozessoren" (*Embedded Processors*) bezeichnet.

Bevor wir uns im Detail mit den Mikrocontrollern beschäftigen, zeigt Bild 2.1-2 ein sehr einfaches eingebettetes System, in dem der Mikrocontroller im wesentlichen die folgenden beiden Funktionen erfüllen muß:

- Er ist für das möglichst schonende Aufladen der wiederaufladbaren Batterie (Akku) zur Spannungsversorgung des Geräts zuständig,
- Er verwaltet die ‚Benutzerschnittstelle', die hier lediglich aus einem Ein/Ausschalter für den Motor des Gerätes sowie eine Reihe von Leuchtdioden (LEDs) zur Anzeige des Betriebszustands ‚ein/aus' des Gerätes und des Ladezustands des Akkus besteht.

[1] nicht ganz korrekt für ‚Anwendungen mit eingebetteter Steuerung'

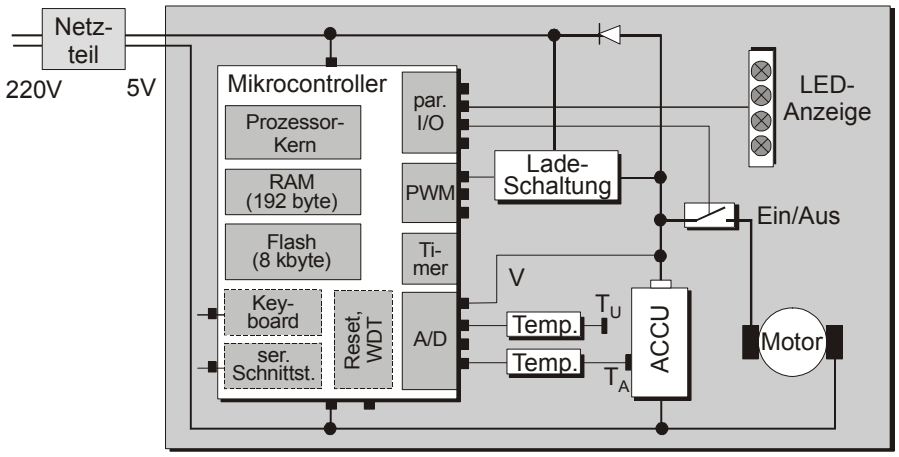

Bild 2.1-2: Ein einfaches Mikrocontroller-System

Zur Feststellung des aktuellen Ladezustands ermittelt der Mikrocontroller einerseits die Akku-Spannung (V), andererseits die Betriebstemperatur des Akkus (T_A). Diese vergleicht er mit der Umgebungstemperatur im Gerät (T_U). Im Bild 2.1-3 ist der Verlauf beider Größen beim Ladevorgang skizziert.

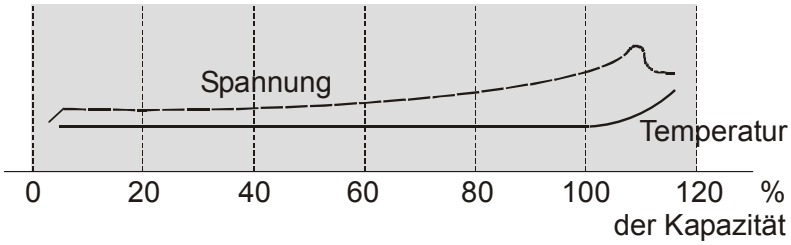

Bild 2.1-3: Spannungs- und Temperaturverlauf beim Ladevorgang

Um eine irreversible Schädigung des Akkus zu verhindern, muß der Mikrocontroller durch eine geeignete Steuerung des Ladevorgangs unbedingt verhindern, daß das lokale Maximum der Akku-Spannung (bei ca. 110% der Kapazität) erreicht oder überschritten wird. Interessanterweise ist der Verlauf der Temperaturmeßlinie, die sich aus der Differenz der Akku-Temperatur und der Umgebungstemperatur errechnet, aussagekräftiger für den Ladezustand als der Spannungsverlauf. Mit den ermittelten Werten steuert der Mikrocontroller einen Ladevorgang, wie er im Bild 2.1-4 skizziert ist.

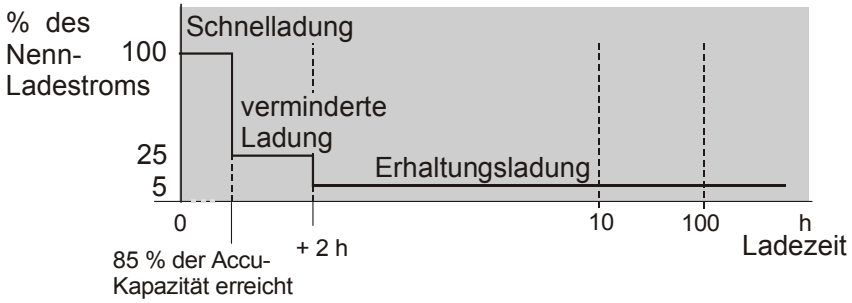

Bild 2.1-4: Verlauf des Ladevorgangs

- Zunächst wird eine Schnelladung mit 100% des Nennladestroms durchgeführt, bis ca. 85% der Akku-Kapazität erreicht ist.
- Daran schließt sich eine maximal 2-stündige Phase der verminderten Ladung mit 25% des Nennladestroms an, wobei stets überprüft wird, daß keine Überladung (über 100%) auftritt.
- Im Anschluß daran schaltet der Mikrocontroller auf die sog. Erhaltungsladung, die mit 5% des Nennladestroms ausgeführt wird und dafür sorgt, daß der Akku stets voll aufgeladen ist.

Die für die beschriebene Aufgabe benötigten Mikrocontroller-Komponenten und Parameter sind im Bild 2.1-2 eingezeichnet. Der verwendete Mikrocontroller ist der 8-bit-Mikroprozessor[2] 68HC908KX8 der Firma Motorola (heute: Freescale), der mit 4 – 8 MHz betrieben wird und in einem kostengünstigen Gehäuse mit 16 Anschlüssen (*Pins*) untergebracht ist. Für die Steuerung des Ladestroms wird ein PWM-Kanal (Pulsweiten-Modulation) eingesetzt. Die Ermittlung der Akku-Spannung und der Temperaturen geschieht über drei Analog/Digital-Wandler-Eingänge. Die zeitliche Überwachung des Ladevorgangs wird über einen Zeitgeber/Zähler-Baustein (*Timer*) vorgenommen. Über eine Leitung der Parallelschnittstelle (*Parallel I/O*) wird der Zustand des Ein/Ausschalters festgestellt; und über weitere Ausgangsleitungen dieser Schnittstelle werden die Leuchtdioden betrieben. Alle erwähnten Komponenten werden im Kapitel 9 ausführlich beschrieben.

Bei dem dargestellten ‚eingebetteten System' handelt es sich übrigens um ein Gerät, das in sehr vielen Haushalten täglich benutzt wird: einen Elektrorasierer mit Akku-Betrieb.

2.2 Mikrocontroller-Eigenschaften und Einsatzgebiete

Einige der folgenden typischen Mikrocontroller-Eigenschaften wurden bereits beschrieben:

- Der Mikrocontroller bearbeitet meist feste Programme für feste Anwendungen.
- Er besitzt einen „nichtflüchtigen" bzw. „Festwert"-Speicher als Programmspeicher, also einen Speicher, der nach dem Abschalten der Betriebsspannung seinen Inhalt nicht verliert.
- Durch die Integration vieler Komponenten auf einem Chip wird der Platzbedarf für die Prozessorsteuerung reduziert. Dadurch kann die Platinengröße verringert werden, oder es können mehr Komponenten auf der Platine untergebracht werden.
- Mikrocontroller haben durch *Power-Down*- bzw. *Sleep*-Modi einen geringeren Stromverbrauch als universelle Mikroprozessoren. Dadurch eignen sie sich besonders für den Batteriebetrieb. Dabei wird immer häufiger eine Spannungsversorgung mit bis zu 3 V oder weniger eingesetzt.

Die bisher beschriebenen Eigenschaften führen zu einer erheblichen Kostenreduktion bei der Baustein- und Platinenherstellung sowie den Betriebskosten:

- In der Entwurfsphase sind schnelle Design- und Programmänderungen möglich, wenn Controller-Versionen mit benutzerprogrammierbaren Festwertspeichern eingesetzt werden.
- Bei der Produktion von großen Stückzahlen werden die Kosten durch „maskenprogrammierte" Versionen verringert, d.h. durch Mikrocontroller, deren Festwertspeicher vom Chip-Hersteller bei der Produktion programmiert werden.

[2] Definition: Das Rechenwerk (für ganze Zahlen) eines **n-bit-Prozessors** kann in einem einzigen Schritt n-bit-Daten verarbeiten.

- Durch die geringe Anzahl externer Komponenten wird die Wartungsfreundlichkeit des Geräts gesteigert.
- Die Betriebssicherheit wird durch die Verringerung der Zahl fehleranfälliger externer Komponenten sowie durch verschiedene Komponenten des Controllers erhöht – wie speziellen Überwachungsschaltungen (*Watch-Dog Timer* – WDT, s. Abschnitt 9.4, KE5), *Power-Down*- bzw. *Sleep*-Modus usw.
- Ein Hardware-Kopierschutz für den Chip-internen Programmspeicher verhindert das unerlaubte Kopieren von kostspielig entwickelten Softwarelösungen.

Wie bereits gesagt, bestehen die Hauptaufgaben und Einsatzarten in der Steuerung und Regelung von Systemen. Dazu kommen in modernen Computersystemen im verstärkten Maß Aufgaben im Zusammenhang mit der Manipulation (z.B. De-/Kompression) und der Übertragung von Daten. In vielen Bereichen stehen die Mikrocontroller dabei in direkter Konkurrenz zu den Digitalen Signalprozessoren (DSP), deren Fähigkeiten zur Lösung der erwähnten Aufgaben stetig gesteigert werden (vgl. Kapitel 3 und 4).

Die Haupteinsatzgebiete für Mikrocontroller sind:

- Auto und Verkehr,
- Konsumelektronik,
- Kommunikation und Datenfernübertragung,
- Computerperipherie,
- Industrieanwendungen,
- SmartCard/Chipkarten-Anwendungen.

Das folgende Bild 2.2-1 zeigt diese Haupteinsatzgebiete und für jedes Gebiet einige wenige Aufgabenbereiche.

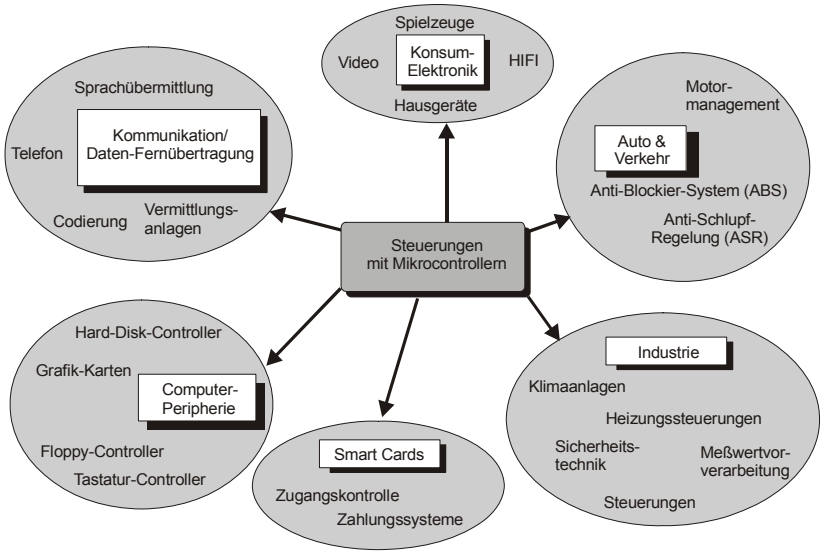

Bild 2.2-1: Haupteinsatzgebiete für Mikrocontroller

Beispielhaft soll hier gezeigt werden, wo im Automobilbereich heute bereits Mikrocontroller (und DSPs) eingesetzt werden (s. Bild 2.2-2). Dies führt dazu, daß in einem Automobil der ‚Oberklasse' bereits einige Dutzend Mikrocontroller ihre Arbeit ausführen.

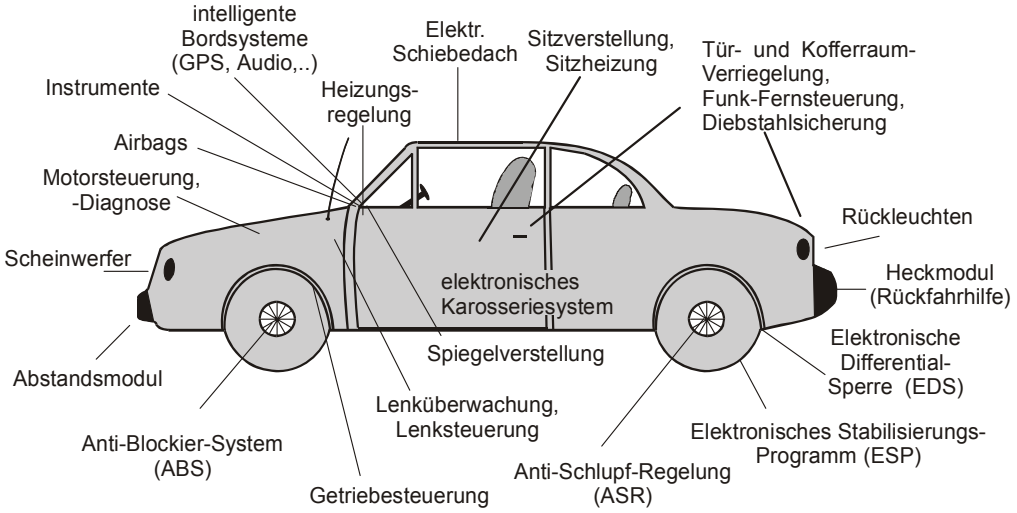

Bild 2.2-2: Einsatz von Mikrocontrollern in einem Automobil

Gruppiert nach verschiedenen Einsatzgebieten ergeben sich die folgenden Einsatz- und Aufgabenbereiche, die wir leider nur aufzählen, aber nicht im einzelnen beschreiben können:

- Antriebssystem: elektronische Motorsteuerung, elektronische Getriebesteuerung, Lambdasonden-Regelung;
- Sicherheitssystem: *Airbags*, Bremsregelung (Anti-Blockier-System – ABS), Anfahrhilfe (Anti-Schlupf-Regelung – ASR), Kurvenstabilisierung (Elektronisches Stabilisations-Programm – ESP), Elektronische Differentialsperre (EDS), Verschleißüberwachung und Diagnosehilfe, Diebstahlsicherung, Wegfahrsperre;
- Lenküberwachung: elektronische Lenkung (*Steering by Wire*);
- Karosseriesystem: Türüberwachung und -verriegelung, Kofferraumverriegelung, Gurtstraffer, Heizung, Klimaanlage, Temperaturregelung, Luftverteilung, Luftdurchsatzregelung, Luftfiltersysteme, Sitzheizung, Sitzverstellung, Standheizung, verstellbare Außenspiegel, Leuchtenüberwachung, Scheinwerferverstellung, Funktionskontrolle;
- Abstandsmodule (vorne und hinten);
- intelligente Bordsysteme: GPS (*Globale Positioning System*), Navigationssysteme, Radio.

Marktprognosen[1] gehen davon aus, daß sich das Marktvolumen für Mikrocontroller im Automobilbereich zwischen 2006 und 2010 um 63% erhöhen wird und dabei der weltweite Umsatz von 5,83 Milliarden Dollar auf 9,52 Milliarden steigen wird.

Die beschriebenen Einsatz- und Aufgabenbereiche stellen sehr unterschiedliche Anforderungen an die Komplexität und Leistungsfähigkeit der verwendeten Mikrocontroller. Anders als im Bereich der universellen Prozessoren werden daher im Mikrocontrollerbereich 4-bit-, 8-bit- und 16-bit-Prozessoren weiterhin in sehr großen Stückzahlen eingesetzt. Nach Umsatz gerechnet, machen diese immer noch bis zu 67% aus. Wegen ihrer erheblich niedrigeren Stückpreise sind also mehr als 2/3 aller verkauften Mikrocontroller aus dieser Klasse. Bild 2.2-3 zeigt die prognostizierten Anteile der verschiedenen Mikrocontroller-Typen bis zum Jahr 2010[2].

[1] Nach einer Studie von Frost & Sullivan, *Embedded Systems Design Europe*, 11/12.07
[2] Quelle: Embedded Systems Design Europe, Mai 2007, S. 14

Nach den eben genannten Prognosen wird für den 32-bit-Markt bis 2011 eine jährliche Wachstumsrate der Stückzahlen von 24,5% erwartet, die – wegen der fallenden Stückpreise – einem jährlichen Umsatzplus von 17,8% entsprechen soll. Der weltweite Gesamtumsatz in diesem Markt betrug 2007 ca. 4,2 Milliarden Dollar. Im selben Jahr hatten die führenden Firmen auf dem 32-bit-Markt die folgenden Anteile: Renesas (Ausgliederung des Halbleiterbereichs von Hitachi und Mitsubishi): 27,7%, NEC (*Nippon Electronics Corporation*): 22,9% und Freescale (ehemals Motorola): 17,3%.

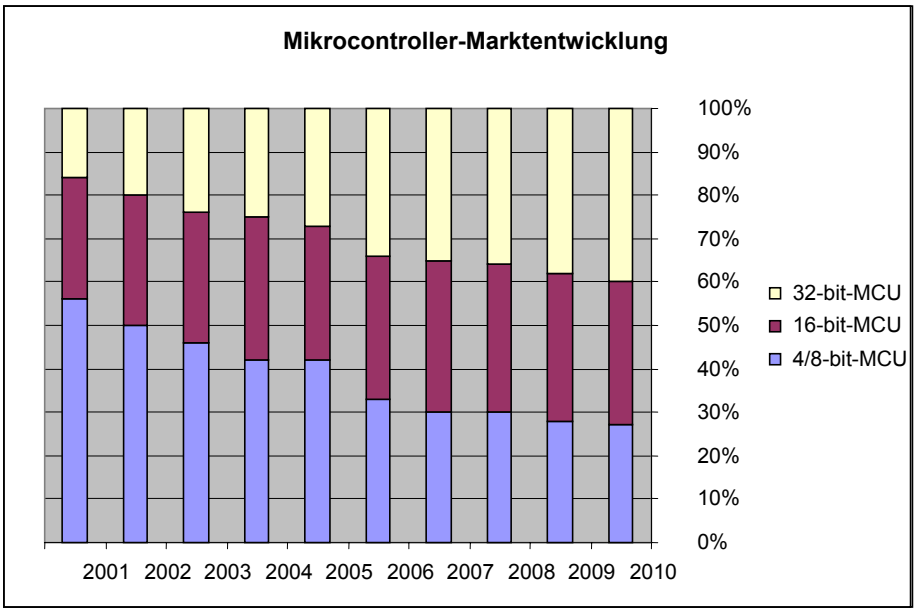

Bild 2.2-3: Marktanteile der Mikrocontroller-Typen

Eine immer größere Bedeutung finden Mikrocontroller auch in Schaltungen, die vom Systementwickler für spezielle Einsatzorte und -zwecke selbst erstellt („programmiert“) werden können, den sog. *Field Programmable Gate Arrays* (FPGAs). Dazu kann der Entwickler die hardwaremäßige Beschreibung eines Mikrocontrollers – ähnlich wie ein Unterprogramm einer höheren Programmiersprache – laden und in der FPGA-Schaltung „einprogrammieren“. Bild 2.2-4 zeigt, daß der Anteil der FPGAs mit integriertem Mikrocontroller in naher Zukunft auf über 40 % steigen wird.

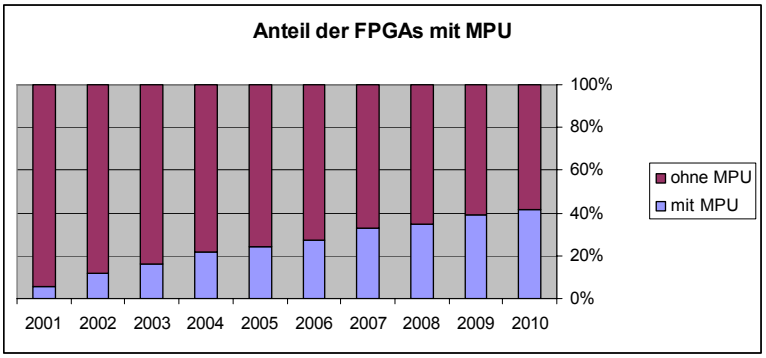

Bild 2.2-4: Marktentwicklung der FPGAs mit Mikrocontroller

2.3 Typischer Aufbau eines Mikrocontrollers

2.3.1 Beschreibung der Komponenten

Im Bild 2.3-1 ist der grobe Aufbau eines Mikrocontrollers skizziert. Danach besteht er aus den folgenden Komponenten bzw. Komponentengruppen:

- dem Prozessorkern,
- eventuellen Erweiterungen des Prozessorkerns für spezielle Berechnungen,
- Programm- und Datenspeichern,
- Standardperipherie-Modulen,
- anwendungsspezifischen Modulen.

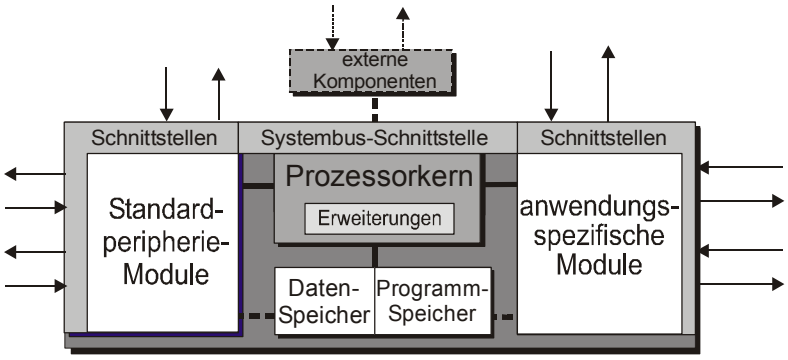

Bild **2.3-1:** Grober Aufbau eines Mikrocontrollers

Der Prozessorkern, auch CPU (*Central Processing Unit*) genannt, kann eine spezielle Entwicklung für einen bestimmten Mikrocontroller bzw. eine Mikrocontroller-Familie sein. Im großen Maße werden jedoch bereits verbreitete universelle Mikroprozessoren als Prozessorkern (*Processor Core*) in Mikrocontrollern eingesetzt. Dadurch wird insbesondere die Softwareentwicklung erleichtert, da auf die Erfahrung von Programmierern und eventuell bereits vorhandene Entwicklungshilfsmittel zurückgegriffen werden kann. Für spezielle Anwendungen erweitert der Hersteller diese universellen Prozessorkerne um weitere Operationswerke, die entweder in den Prozessorkern selbst oder als eigenständiges Rechenwerk ‚neben' dem Prozessorkern auf dem Chip integriert werden. Zum Teil werden diese Erweiterungen in alle Mitglieder einer Controllerfamilie eingebaut, häufig aber aus Kostengründen nur in spezielle Familienprodukte, so daß der Anwender die für ihn geeignete ‚Controller-Ausstattung' wählen kann.

Programme und Daten können dabei in einem gemeinsamen Speicher oder in getrennten Speichern abgelegt werden. Standardperipherie- und anwendungsspezifische Module wirken entweder nur im ‚Inneren' des Mikrocontrollers oder besitzen eine Schnittstelle zur ‚Außenwelt'. Welche Komponenten zu diesen Modulgruppen gehören, werden wir weiter unten erklären. Über die (eventuell vorhandene) Systembusschnittstelle kann der Mikrocontroller extern meist um weitere Komponenten ergänzt werden, insbesondere auch um eine oder mehrere Speichereinheiten oder einen Coprozessor.

Im nächsten Bild 2.3-2 stellen wir den inneren Aufbau eines Mikrocontrollers in feinerer ‚Auflösung' dar. Darin haben wir alle wesentlichen Module zusammengefaßt, die in gängigen

Mikrocontroller-Familien zu finden sind. Reale Controller verfügen häufig nicht über alle Komponenten, sondern nur über eine mehr oder weniger große Auswahl daraus. Andererseits gibt es Controller, die über zusätzliche spezifische Komponenten verfügen, die wir in unseren allgemeinen Überblick jedoch nicht aufnehmen konnten.

Bei einfacheren Mikrocontrollern werden die Komponenten z.T. direkt mit den entsprechenden Signalen des Prozessorkerns verbunden. Hersteller von höherwertigen Controllern, die ihre Produkte in einer oder mehreren Controller-Familien mit einer weiten Palette von unterschiedlichen Ausstattungen an Peripheriekomponenten anbieten, setzen meist einen firmenspezifischen, schnellen ‚Peripheriebus' in ihren Controllern ein. Diese werden z.B. als *Inter-Module Bus* (IMB, Motorola/Freescale) oder *Flexible Peripheral Interconnect* (FPI, Infineon) bezeichnet. Der Einsatz dieser Busse erleichtert insbesondere den modularen Aufbau von Controllern aus einer Bibliothek von Peripheriekomponenten mit einheitlichen Busschnittstellen. Um die Kommunikation zwischen Prozessorkern und Speicher nicht durch Buszugriffe der Komponenten zu belasten, besitzen viele Controller einen eigenen Prozessor-/Speicherbus, der mit dem Peripheriebus über eine „Brücke" verbunden ist. So sind simultane Übertragungen auf beiden Bussen möglich.

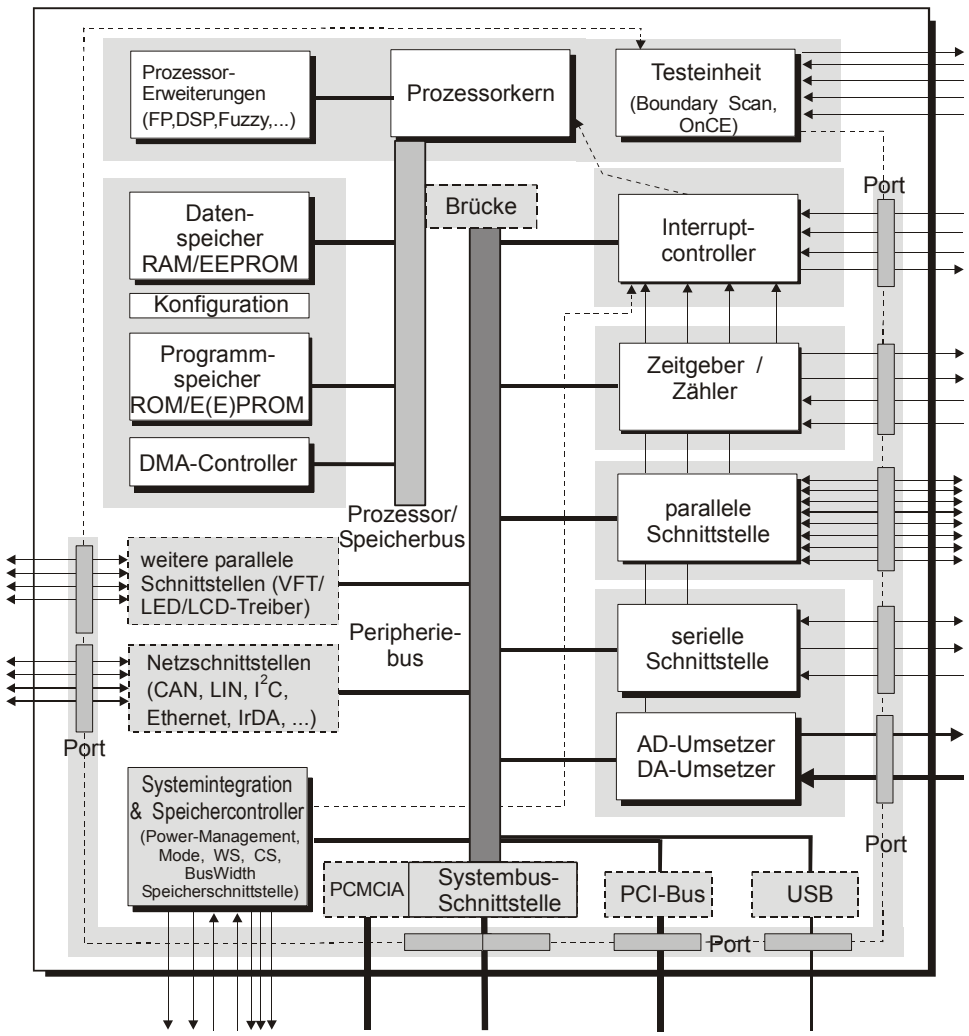

Bild 2.3-2: Feinstruktur eines komplexen Mikrocontrollers

Im folgenden werden wir die im Bild 2.3-2 dargestellten Komponenten kurz beschreiben. Die ausführliche Beschreibung der Komponenten folgt in den späteren Kapiteln.

- Der **Prozessorkern** kann als CISC- oder RISC-Prozessor realisiert sein. Er enthält ein Integer-Rechenwerk mit 4, 8, 16 oder 32 bit Verarbeitungsbreite. Zum Rechenwerk gehören häufig ein schnelles Parallel-Multiplikationswerk und ein schnelles Divisionswerk. Die Rechenwerke von leistungsfähigen Controllern verfügen meist über einen großen Registersatz für Adressen und Daten.
- Als **Prozessor-Erweiterungen** werden im verstärkten Maße Multiplizier-Akkumulier-Rechenwerke (*Multiply-Accumulate* – MAC) oder komplexere digitale Signalverarbeitungseinheiten[1] eingesetzt. Weiterhin finden sich Gleitpunkt-Arithmetikeinheiten (*Floating-Point Unit* – FPU) – sofern sie nicht schon Bestandteil des universellen Prozessorkerns sind –, Graphik-Einheiten oder *Fuzzy-Logic*-Einheiten[2] (zur Berechnung von Algorithmen der ‚unscharfen' Logik). Wenn diese Erweiterungen durch besondere Befehle im Befehlssatz des Prozessors programmiert werden können, so spricht man auch von **Coprozessoren**.
- Zu den Prozessorerweiterungen kann man auch die **Testeinheit** zählen, die bei modernen, höherwertigen Controllern zur ‚Grundausstattung' gehört. Sie kann häufig in zwei verschiedenen Betriebsmodi arbeiten: als sogenannter JTAG-Port (*Joint Test Action Group*) erlaubt sie in komplexen Systemen, die Verbindungsleitungen zwischen den Bausteinen zu testen. Dieser Test wird als ***Boundary Scan*** bezeichnet. In der zweiten Betriebsart ermöglicht die Testeinheit, in der Entwicklungsphase eines Systems die korrekte Funktion des Controllers und seiner Peripherie zu überprüfen, eventuelle Fehler aufzuspüren und zu lokalisieren. Diese Funktion wird ***On-Chip Emulation*** (OnCE) genannt. Auf beide Betriebsmodi der Testeinheit gehen wir im Anhang dieser Kurseinheit genauer ein.
- Die **Speichereinheit** enthält zunächst den Programmspeicher, der bis zu 2 Mbyte groß sein kann und meist aus benutzerprogrammierbaren Festwertspeicherzellen[3] besteht. Häufig kann der Programmspeicher durch das „Durchbrennen" einer Schmelzsicherung oder aber reversible interne Maßnahmen gegen unerlaubtes Auslesen geschützt werden. Der Programmspeicher kann extern bei einfachsten Controllern bis auf 64 kbyte, bei Hochleistungscontrollern bis auf 4 Gbyte erweitert werden. Der Maximalwert für den internen Datenspeicher, der aus statischen Schreib-Lese-Speicherzellen (*Static Random Access Memory* – SRAM) oder Festwert-Speicherzellen (EEPROM-Zellen) besteht, liegt bei 128 kbyte. Einfache Controller müssen sich häufig aber mit sehr wenigen Schreib-Lese-Speicherzellen (ein oder wenige kbyte) zufrieden geben.

  Im Adreßbereich des Datenspeichers findet sich gewöhnlich auch die sog. I/O-Page, in der alle Steuer- und Statusregister (*Special Function Registers* – SFR) der internen Controller-Komponenten zusammengefaßt werden (s. Unterabschnitt 8.2.4, KE4). Die Anzahl dieser Spezialregister kann bei komplexen Controllern bei einigen Hundert liegen.

  Bei einigen Mikrocontrollern ist der gesamte Datenspeicher oder ein bestimmter Teil davon ‚bitweise' ansprechbar, d.h. die kleinste adressierbare und manipulierbare Einheit ist hier das

[1] mit Hardware-Schleifensteuerung, Daten-Adreßgeneratoren usw., s. Unterabschnitte 5.2.5 und 5. 4.3, KE2.

[2] Fuzzy-Logik (englisch: *fuzzy* = ungenau, verschwommen, unscharf): Verallgemeinerung der zweiwertigen (Boole'schen) Logik, läßt auch Wahrheitswerte zwischen WAHR und FALSCH zu. Damit sind auch unscharfe Mengenabgrenzungen mathematisch behandelbar.

[3] Heute oft EEPROM (Electrically Erasable Programmable Read-Only Memory) oder Flash-EEPROM.

einzelne Bit – und nicht ein ganzes Byte. Diese Adressierungsmöglichkeit unterstützt hardwaremäßig die Bit-Manipulationsbefehle, auf deren Bedeutung für Mikrocontroller wir weiter unten eingehen werden. Nicht zuletzt ermöglichen es diese Befehle, bestimmte Bitfelder der o.g. Spezialregister gezielt zu lesen oder zu verändern.
Mit Hilfe der Speicher-Konfigurationseinheit kann der Entwickler eines Mikrocontrollersystems festlegen, in welchem Bereich des Adreßraums der interne Programm- und Datenspeicher liegen sollen. Dazu wird ihm in der Regel eine kleinere Anzahl von Auswahlmöglichkeiten zur Verfügung gestellt, die den Systementwurf stark erleichtern können. Darüber hinaus kann er entscheiden, ob der Controller im sogenannten Mikroprozessor- oder Mikrocomputer-Modus arbeiten soll: Im ersten Fall ist er um externe Speichermodule erweitert, im zweiten Fall muß er mit den internen Speicherbereichen auskommen. Die Auswahl zwischen beiden Modi geschieht z.B. dadurch, daß der Controller (nur) während der Hardware-Initialisierung nach einem Rücksetzen (*Reset*) den Zustand bestimmter Busleitungen abfragt, die vom Systementwickler geeignet beschaltet wurden.

- Wegen seiner Funktion des ‚direkten Speicherzugriffs' zählen wir zur Speichereinheit auch einen eventuell vorhandenen **DMA-Controller** (*Direct Memory Access*, s. Abschnitt 9.3, KE5), der es den Peripherie-Komponenten erlaubt, selbstständig – d.h. ohne Unterstützung durch den Prozessorkern – auf den Speicher zuzugreifen. Dieser besitzt in der Regel mehrere ‚Kanäle' – typischerweise 6, maximal bis zu 32 – die unabhängige, nach Prioritäten gesteuerte Zugriffe der Peripheriekomponenten auf den Speicher ermöglichen.

Die nun beschriebenen Komponenten bezeichnen wir als Standardperipherie-Module, da sie bei den meisten Mikrocontrollern zu finden sind:

- Das **Zeitgeber/Zähler-Modul** (*Timer*), das im Abschnitt 9.4, KE5, ausführlich beschrieben wird, erzeugt eine ganze Palette von Zeitfunktionen für Prozeßsteuer-Aufgaben. Im wesentlichen besteht ein Timer aus einem Binärzähler, der von einem einstellbaren Anfangswert auf ‚0' herunterzählt. Zu den erzeugten Funktionen gehören insbesondere das Zählen und Erfassen („Auffangen") externer Ereignisse (*Input Capture*) sowie die Erzeugung externer Ereignisse (*Output Compare*). Weitere wichtige Funktionen des Zeitgeber/Zähler-Moduls sind der Einsatz als *Watch-Dog Timer* (WDT) zur Überwachung des Prozessorzustands und zur Einstellung eines unkritischen Zustands nach dem Auftreten eines Fehlers sowie die Verwendung als „Echtzeituhr" (*Real Time Clock* – RTC), die die Zeit in Stunden, Minuten, Sekunden, Zehntelsekunden angibt. Dieses Modul ist z.T. als eigenständiger (mikro-)programmierbarer Prozessor implementiert (*Time Processing Unit* – TPU) und kann dann bis zu 32 unabhängig arbeitende „Kanäle" zur Verfügung stellen.
- Der **Interrupt-Controller** (s. Abschnitt 9.2, KE5) eines komplexen Mikrocontrollers muß häufig einige Dutzend (im Extremfall bis über 200) „Interruptquellen" verwalten. Das sind System-Komponenten, die Unterbrechungswünsche zur Bearbeitung eigener Aufgaben an den Prozessorkern stellen können. Dabei kann meist zwischen mehreren Prioritätsreihenfolgen gewählt werden – z.B. zwischen der ‚unfairen' Zuteilung fester Prioritäten oder der ‚fairen' *Round-Robin*-Strategie, bei der die höchste Priorität zyklisch rotierend vergeben wird.

- Die Interrupt-Anforderungen können von den integrierten Komponenten oder aber von externen Komponenten an bestimmten Anschlüssen gestellt werden. Dabei ermöglichen es viele Mikrocontroller, nicht benutzte Anschlüsse anderer interner Komponenten zu Interrupt-Leitungen „umzuprogrammieren", die dann selektiv aktiviert bzw. deaktiviert werden können.
- Zu den digitalen Ein-/Ausgabe-Ports (I/O-Ports) gehören einerseits die parallelen Schnittstellen, die bis zu einigen Dutzend programmierbare Ein-/Ausgabe-Leitungen sowie spezielle Steuer- und Kontroll-Leitungen zum ‚Quittungsaustausch' (Handshake) umfassen. In vielen Anwendungen werden diese Leitungen nicht zur parallelen Übertragung von Mehr-Bit-Werten, sondern zur Ausgabe von einzelnen Steuersignalen bzw. zum Einlesen einzelner Zustandssignale verwendet. Deshalb werden sie auch als Allzweck-Ein-/Ausgabesignale bezeichnet (*General Purpose I/O* – GPIO).
  Häufig sind zusätzlich die Anschlußsignale nicht benötigter interner Komponenten als parallele Ein-/Ausgabeleitungen konfigurierbar, was im Bild 2.3-2 durch eine graue Unterlegung gekennzeichnet sein soll. Dies gilt bei einem im Mikrorechner-Modus arbeitenden Controller oft sogar für die – aus externem Adreß-, Daten- und Steuerbus bestehende – Systembusschnittstelle.
  Zu den digitalen Ein-/Ausgabeports zählen weiterhin die asynchronen und synchronen seriellen Schnittstellen. Unter diesen sind die asynchrone V.24-Schnittstelle (RS 232) sowie die synchrone SPI-Schnittstelle (*Serial Peripheral Interface*) besonders wichtig, die im Abschnitt 9.7, KE6, beschrieben werden.
- Die **analogen Ein-/Ausgabe-Ports** bestehen in der Regel nur aus Analog/Digital-Umsetzer (ADU, auch: A/D-Wandler, *A/D Converter* – ADC), welche die Digitalisierung analoger Signale bis zu einer Auflösung von 10 bit – in Ausnahmefällen auch 12 oder 16 bit – ermöglichen. Dazu kann der Mikrocontroller bis zu 40 analoge Eingänge besitzen, von denen jeweils einer über einen Analog-Multiplexer auf den A/D-Wandler gegeben wird. Durch einen zuschaltbaren Abtast- und Halteverstärker (*Sample and Hold* – S&H) kann das analoge Signal für die Dauer einer Umwandlung konstant gehalten werden. Zu den analogen Eingabequellen zählen immer häufiger auf dem Chip integrierte, mit eigenem A/D-Wandler ausgestattete Temperatursensoren, häufig auch mit Thermostatfunktion zur Überwachung eines zulässigen Temperaturbereichs.
  Seltener findet man bei Mikrocontrollern integrierte Digital/Analog-Umsetzer (DAU, auch: D/A-Wandler, *D/A Converter* – DAC). Wie die A/D-Wandler besitzen sie eine maximale Auflösung von 10 bit. Häufiger wird zur D/A-Wandlung – wie im Unterabschnitt 9.4.4, KE5, beschrieben – ein PWM-Signal ausgegeben, das durch ein externes Integrierglied (Integrator) in einen analogen Wert umgewandelt wird.
  A/D- und D/A-Wandler-Einheiten für Mikrocontroller werden im Abschnitt 9.8 beschrieben.
- Das **Modul zur Systemintegration** (*System Integration Module* – SIM) besteht hauptsächlich aus einem Systembus-Controller, der die Erzeugung von Signalen zur Auswahl verschiedener Speicher/Register-Bereiche (*Chip Select* – CS), zur Steuerung der Datenbusbreite und zur Vergabe des Buszugriffs (Arbitrierung) sowie das Einfügen von Wartezyklen (*Wait States*) zur Aufgabe hat (s. Abschnitt 8.2, KE4).
  Außerdem enthält das Modul häufig einen Speichercontroller, der die Steuersignale für den

[4] Als Port wird allgemein eine Gruppe von Anschlüssen bezeichnet, die in einem funktionalen Zusammenhang stehen.

Einsatz unterschiedlichster Schreib/Lese- oder Festwert-Speicherbausteine generiert. Eine weitere Komponente übernimmt die Steuerung der Leistungsaufnahme (*Power Management*): Sie versetzt die Komponenten des Controllers in verschiedene Stromspar-Modi, die sich nach der Frequenz ihres Arbeitstakts und der Höhe der Versorgungsspannung unterscheiden (vgl. Abschnitt 2.3, KE1).

Die folgenden Komponenten sind nur in Mikrocontrollern für besondere Anwendungen integriert. Häufig existieren in umfangreichen ‚universellen' Controller-Familien spezielle Produkte, die über diese Komponenten verfügen.

- Zum Anschluß von Ein-/Ausgabegeräten für den Benutzer verfügen diese Mikrocontroller über weitere **anwendungsspezifische parallele Schnittstellen**. Dazu gehören Ausgänge, die hohe Ströme für den Anschluß von Leuchtdioden-Anzeigen (*LED Displays*) bzw. hohe Spannungen (bis 40 V) für den Anschluß von Vakuum-Fluoreszenz-Röhren (*Vacuum Flourescent Tube* – VFT) liefern. Für den Einsatz von ein- oder mehrfarbigen Flüssigkristall-Anzeigen (*Liquid Crystal Display* – LCD) mit bis zu 160 getrennt ansteuerbaren Anzeigepunkten (Segmenten) gibt es Ausgänge, die hohe digitale Wechselspannungen erzeugen. Weitere spezielle Schnittstellen, die ebenfalls Treiber für relativ hohe Ströme zur Verfügung stellen, dienen zur Ausgabe von Signalen für die Ansteuerung von Schrittmotoren.
  Als spezielle Eingabemodule findet man Schnittstellen zum Anschluß von grau oder farbig darstellenden, berührungsaktiven Anzeigen (*Touch Screens*) sowie von kleinen Tastaturen mit Zeilen/Spalten-Multiplexansteuerung (*Keypads*).
- Als anwendungsspezifische serielle Schnittstelle wird die **Infrarot-Schnittstelle** immer wichtiger, die mit Hilfe von infrarotem Licht serielle Daten mit einer Bitrate von 115 kbit/s bis 4 Mbit/s nach dem standardisiertem IrDA-Protokoll (*Infrared Data Association*) synchron überträgt.
- In komplexen Systemen wird immer häufiger eine größere Zahl von Mikrocontrollern verwendet, die untereinander Daten austauschen müssen. Zu ihrer Kopplung verfügen viele Controller über sogenannten **Netzschnittstellen**. Eine der erfolgreichsten Schnittstellen dieser Art ist der **CAN-Bus** (*Controller Area Network*), der insbesondere im Automobilbau und im industriellen Bereich eingesetzt wird. Ihn werden wir im Abschnitt 8.4, KE4, ausführlich beschreiben.
  Eine weitere, etwas leistungsschwächere serielle Netzschnittstelle, der **I2C-Bus** (*Inter-Integrated Chip Bus*), wurde bereits vor Jahrzehnten von der Firma Philips zur Verbindung von hochintegrierten Bausteinen entwickelt – nicht nur zur Kopplung von Mikrocontrollern. Beim I^2C-Bus handelt es sich ebenfalls um eine bidirektionale 2-Draht-Schnittstelle mit Übertragungsraten zwischen 100 und 400 kbit/s. Er hat in den letzten Jahren insbesondere im Bereich der SmartCards eine große Bedeutung erlangt – also bei den weitverbreiteten Chipkarten, deren integrierter ‚Chip' ein Mikrocontroller ist.
- Mit der fortschreitenden Miniaturisierung der universellen ‚persönlichen Rechnersysteme', also z.B. der PDAs (*Personal Digital Assistants*), Pocket-PCs und Laptops, wird die Rolle der Mikrocontroller in diesem Bereich immer wichtiger. Deshalb werden ihre leistungsfähigsten Vertreter im verstärkten Maß mit denjenigen Busschnittstellen und den dazu erforderlichen Steuermodulen ausgerüstet, die in diesem Mikrorechner-Bereich weit verbreitet sind. Dazu gehören insbesondere der **PCI-Bus** (*Peripheral Component Interconnect*) – mit 32 bit

Busbreite und 33 MHz Bustakt – sowie der **USB** (*Universal Serial Bus*). Im Rahmen dieses Kurses werden wir nur auf den „moderneren“ USB eingehen (vgl. Abschnitt 8.3, KE4). Häufig wird in den eben erwähnten Mikrocontrollern auch die im Laptop-Bereich verwendete **PCMCIA-Schnittstelle** (*Personal Computer Memory-Card International Association*) oder ihr Nachfolger, das *PC-Card-Interface* integriert. Die zuerst genannte Schnittstelle ist ein 16-bit-Derivat des ISA-Busses, die zweite ein 32-bit-Derivat des PCI-Busses. Mit Hilfe von genormten, flachen Einschubkarten erlauben es diese Schnittstellen, den Rechner um verschiedene externe Komponenten zu erweitern – z.B. um ein externes Arbeitsspeicher-Modul oder ein Modem (Modulator/Demodulator) zum Anschluß des Rechners an das Telefonnetz.

Das folgende Bild 2.3-3 zeigt das Photo eines Mikrocontroller-Chips, englisch auch *Die* („Mikroplättchen“) genannt. Dabei handelt es sich um den (relativ) einfachen 8-bit-Controller MC68HC908 der Firma Freescale. Durch die eingezeichneten Umrandungen wird die Lage der einzelnen Komponenten dargestellt. Das Bild macht sehr gut deutlich, daß der eigentliche Prozessor, die CPU, mit ca. 4,3% nur noch sehr wenig Platz beansprucht. Eine immer größere Fläche nimmt bei modernen Mikrocontrollern hingegen der interne Speicher ein, im Bild mit RAM (*Random Access Memory* – Schreib-/Lesespeicher), ROM (*Read-Only Memory* – Festwertspeicher) und *Flash* (veränderbarer Festwertspeicher, zuzüglich der Steuerung: *Flash Control*) bezeichnet. Zusammen sind das ungefähr 20% der Chipfläche.

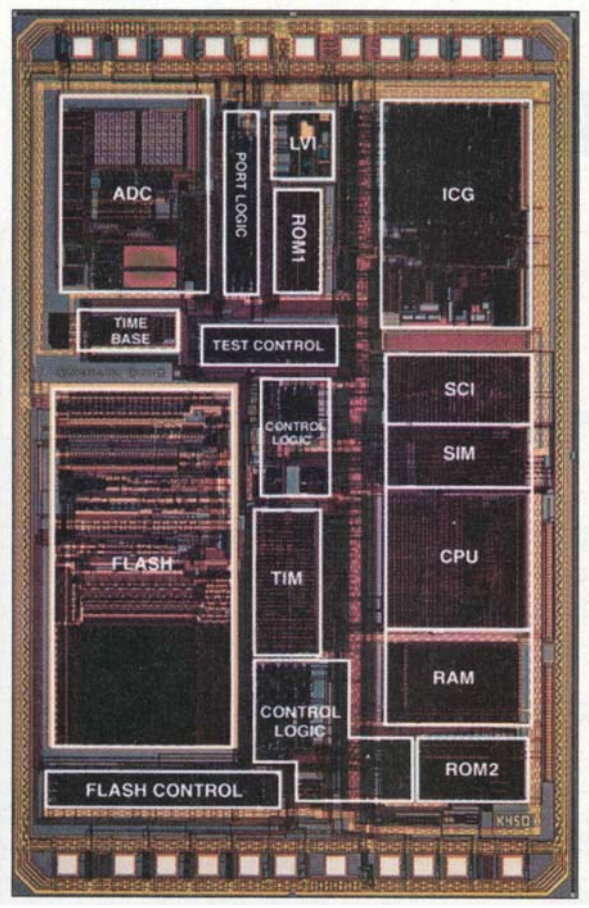

Bild 2.3-3: Chip-Photo eines einfachen Mikrocontrollers

Die weiteren im Bild benutzten Abkürzungen bedeuten:

ICG:	integrierter Taktgenerator (*Internal Clock Generator*)
Control Logic:	Steuerung der Peripheriekomponenten
Time Base:	frei umlaufender Binärzähler zur Vorgabe einer Zeitbasis oder externer Takt,
TIM:	Zeitgeber-/Zähler-Modul (*Timer Interface Modul*),
Port Logic:	Schaltung zur Steuerung der Ein-/Ausgabeleitungen mit Mehrfachbelegung,
ADC:	Analog/Digital-Wandler (*Analog/Digital Converter*),
SCI:	asynchrone serielle Schnittstelle **(***Serial Communications Interface***),**
SIM:	Modul zur Systemintegration (*System Integration Module*)
LVI:	Überwachung des Betriebsspannungs-Pegels (*Low Voltage Inhibit*)
Test Control:	Testschaltung

2.3.2 Steuerung der Leistungsaufnahme

Wie bereits im Abschnitt 1.3 beschrieben wurde, spielt in vielen eingebetteten Systemen die Steuerung und Verringerung der Leistungsaufnahme (*Power Management*) eine sehr wichtige Rolle. Aus Kosten- und Platzgründen verbietet sich z.B. der Einsatz eines Lüfters oder großer Kühlkörper zur Abführung der Wärme. Außerdem sind viele dieser Systeme für den Batteriebetrieb ausgelegt. Hier kommt es auf eine möglichst effektive Nutzung der Batteriekapazität an. Mikrocontroller können daher häufig in einen von mehreren Betriebsmodi (*Low-Power Modes*) versetzt werden, in denen die Leistungsaufnahme – unterschiedlich stark – reduziert wird. Der Übergang aus dem ‚Normalbetrieb', in dem Prozessorkern und Komponenten mit voller Taktgeschwindigkeit und hoher Betriebsspannung arbeiten, in einen dieser Zustände wird durch spezielle Befehle oder aber durch das Setzen bestimmter Bits in den Steuerregistern veranlaßt. Die Komponente des Controllers, welche die verschiedenen Modi ‚verwaltet', wird als *Power Management Module* bezeichnet.

- Im ***Stop Mode*** wird der Controller – mit dem Taktgenerator und allen internen Komponenten – angehalten und seine Ausgänge werden hochohmig geschaltet. Ein entsprechender Assemblerbefehl kann z.B. LPSTOP (*Low-Power Stop*) heißen.
- Im ***Power-Down Mode*** wird der Controller ebenfalls vollständig angehalten; zusätzlich kann seine Betriebsspannung aber noch auf einen Wert herabgesetzt werden[5], der zur Erhaltung der in den internen RAM-Speichern abgelegten Informationen gerade noch ausreicht. Ein Befehl zum Wechsel in den *Power-Down Mode* kann z.B. PWRDN heißen.
- Im ***Slow Mode*** wird die Taktfrequenz des gesamten Controllers um einen Faktor reduziert, der häufig im auslösenden Befehl angegeben werden kann. Mögliche Faktoren sind z.B. 16, 32, 64, 128.
- Im ***Idle Mode*** (auch *Wait Mode* genannt) wird nur der Takt vom Prozessorkern getrennt sowie ein eventuell vorhandener *Watch-Dog Timer* angehalten; die Peripheriekomponenten können jedoch mit voller Taktgeschwindigkeit weiterarbeiten. Häufig ist es möglich, einige oder alle Komponenten durch bestimmte Bits ihrer Steuerregister ebenfalls abzuschalten. Ein Befehl zum Wechsel in den *Idle Mode* heißt z.B. IDLE.
- Im ***Sleep Mode*** wird ebenfalls der Takt des Prozessorkerns sowie eines eventuell vorhandenen *Watch-Dog Timers* angehalten, die Peripheriekomponenten arbeiten hier nur mit herabgesetzter Taktgeschwindigkeit weiter.

Daneben existieren noch eine ganze Reihe von weiteren Stromspar-Modi, auf die wir hier jedoch nicht näher eingehen wollen. Das ‚Aufwecken' (*Wake Up*) des Controllers aus den beschriebenen Betriebszuständen geschieht durch

- das Ändern bestimmter Bits in einem Steuerregister durch entsprechende Befehle, sofern der Prozessor noch zur Programmbearbeitung in der Lage ist,
- eine externe Interruptanforderung, sofern der Interrupt vor dem Eintritt in den Stromspar-Modus aktiviert wurde,
- die Anforderung einer Einzelschrittausführung (*Trace*),
- oder ein Rücksetzsignal (*Reset*).

Nach dem Aufwecken können bei einigen Modi sehr viele Taktzyklen vergehen, bis der abgeschaltete oder in seiner Frequenz herabgesetzte Taktgenerator wieder stabil arbeitet. Während dieser Zeit kann der Prozessor seine Arbeit noch nicht wieder aufnehmen.

2.3.4 Spezialbefehle bei Mikrocontrollern

In diesem Abschnitt wollen wir uns mit einigen Befehlen und Befehlsgruppen beschäftigen, die wesentlich für die Leistungsfähigkeit eines Mikrocontrollers sind. Den Befehlssatz eines universellen Mikroprozessors werden wir im Abschnitt 6.2, KE3, beschreiben. (Bei den im Abschnitt 1.2 erwähnten Mischformen aus µC und DSP kommen dazu noch die typischen DSP-Befehle nach Unterabschnitt 6.2.4, KE3).

2.3.4.1 Bitmanipulations- und Bitfeld-Befehle[6]

- **Bitmanipulationsbefehle** erlauben es, ein einzelnes Bit in einem Speicherwort, einem Prozessorregister oder einem Spezialregister (*Special Function Register* – SFR) gezielt zu testen, d.h. auf den Wert ‚0' oder ‚1' abzufragen. Im Anschluß an diesen Test kann das Bit auf ‚1' oder ‚0' gesetzt werden bzw. invertiert werden. In Mikrocontroller-Anwendungen werden diese Befehle häufig zur Initialisierung und Manipulation von Portleitungen, zur Aktivierung einzelner Steuerleitungen, zur häufigen Abfrage von Statussignalen sowie zur Bestimmung von Interruptquellen benutzt, die über ein Interrupt-Anforderungsregister einen Unterbrechungswunsch stellen.
- **Bitfeld-Testbefehle** vergleichen einen Registerinhalt mit einer im Befehl vorgegebenen ‚Maske', d.h. mit einer beliebigen 0-1-Bitkombination. Nur wenn das Register an allen Bitpositionen, an der die Maske eine ‚1' hat, mit dem geforderten Wert ‚0' (*Low*) bzw. ‚1' (*High*) übereinstimmt, wird in einem speziellen Register des Prozessors, dem Statusregister, das sogenannte Nullbit[7] (*Zero Flag*) gesetzt.
- Kombinierte **Bitfeld-Test- und Manipulations-Befehle** testen zunächst die durch die Maske definierten Bits auf ‚0' bzw. ‚1' und setzen ggf. das oben genannte Nullbit. Unabhängig vom Wert des Nullbits invertieren sie danach die durch die Maske selektierten Bits oder setzen sie auf ‚0' bzw. ‚1'.

---

[5] Eine Halbierung der Betriebsspannung bedeutet in etwa eine Viertelung der Leistungsaufnahme.
[6] s. Unterabschnitt 6.2.4, KE3..
[7] Einige Prozessoren verwenden dazu das Übertragsbit (*Carry Flag*) im Statusregister, s. Abschnitt 5.5, KE2.

- Kombinierte **Bitfeld-Test- und Verzweigungs-Befehle** führen zunächst den eben beschriebenen Test durch und setzen ggf. das Nullbit. Falls das Bit gesetzt wird, führen sie danach eine Programmverzweigung (*Branch*) durch. Wird das Nullbit nicht gesetzt, verhalten sie sich wie ein einfacher NOP-Befehl (*No Operation*), der keinerlei sonstige Funktion ausführt.

2.3.4.2 Tabellensuch- und Interpolationsbefehl

Der Tabellensuch-Befehl (*Table Look-Up* – TBL) ist sehr wichtig in Steueranwendungen, in denen wiederholt mit Kennlinienfeldern gearbeitet wird. Er ermöglicht die Speicherung einer reduzierten Anzahl von Datenwerten in einer Tabelle und berechnet daraus – durch lineare Interpolation – alle Zwischenwerte. Als Anwendungsbeispiel sei hier eine Motorsteuerung genannt, bei der die aktuelle Geschwindigkeit oder das Drehmoment als Eingangsgröße in einer Tabelle abgelegt werden und daraus der erforderliche Zündwinkel als Ausgangsgröße berechnet wird. In dem folgenden Exkurs beschreiben wir die Ausführung des TBL-Befehls genauer.

Exkurs: Ausführung des TBL-Befehls

Die Basisdaten für den TBL-Befehl[8] werden in einer Tabelle mit n Tupeln (X^i, Y^i) im Speicher abgelegt. Die Ordinatenwerte Y^i können wahlweise Bytes, 16-bit-Wörter oder 32-bit-Doppelwörter sein. Die Abszissenwerte X^i sind als 16-bit-Festpunktzahlen im Format (8.8) – d.h. 8 Stellen vor dem Punkt, 8 Stellen dahinter – vorgegeben, der Dezimalpunkt liegt also zwischen Bit 7 und Bit 8. Der gebrochene Anteil (*Fractional*) der Abszissenwerte X^i ist auf ‚0' gesetzt, d.h. die Abszissenwerte sind ganze Zahlen: $X^i = X_7 \ldots X_0 \,.\, X_{-1} \ldots X_{-8} = X_7 \ldots X_0 \,.\, 0 \ldots 0$. Somit kann die Tabelle maximal 256 Tupel enthalten. Die (ganzzahligen) Abszissenwerte X^i werden mit der Länge der Y-Daten skaliert (× 1, 2, 4 byte) und als relative Adressen (‚Index') für den Zugriff auf die gespeicherte Tabelle verwendet. Unter diesen Adressen werden die zugehörigen Ordinatenwerte Y^i der Datentupel abgelegt.

Im Bild 2.3-4 ist die Ausführung des TBL-Befehls skizziert.

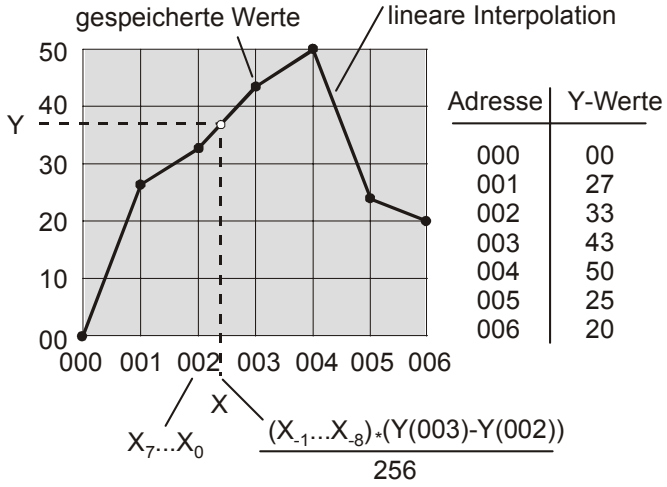

Bild 2.3-4: Funktion des TBL-Befehls

[8] Wir orientieren uns hier an der Mikrocontroller-Familie MC683XX von Motorola/Freescale.

Er ermittelt nach folgendem Verfahren für einen beliebigen Wert

$$X^i = X_7 \ldots X_0 \,.\, X_{-1} \ldots X_{-8}$$

den entsprechenden y-Wert:

- Zunächst wird durch Abschneiden des *Fractional*-Teils von X der (ganzzahlige) Abszissenwert $X^i = X_7 \ldots X_0$ ermittelt.
- Zu den Abszissenwerten X^i, X^{i+1} werden die Ordinatenwerte $Y(X^i)$ und $Y(X^{i+1})$ aus der Tabelle gelesen und ihre Differenz $(Y(X^{i+1}) - Y(X^i))$ berechnet.
- Diese Differenz wird mit dem *Fractional*-Teil $.X_{-1} \ldots X_{-8} = (X_{-1} \ldots X_{-8})/256$ von X multipliziert[9].
- Der so ermittelte Wert wird stellenrichtig, also unter Berücksichtigung der Lage des Punktes, zum Ordinatenwert $Y(X^i)$ hinzuaddiert. Das Ergebnis dieser Addition (u.U. mit einem negativen Summanden) ist der gesuchte Ordinatenwert zu X im Festpunktformat (m.8).

Die weitere Verarbeitung des Ergebnisses hängt vom verwendeten Format des TBL-Befehls ab, der in vier verschiedenen Varianten existiert. Zwei Varianten liefern einen vorzeichenbehafteten bzw. vorzeichenlosen gerundeten Wert (*signed, unsigned*), die beiden anderen Varianten die entsprechenden Werte in nicht gerundeter Form.

In jedem Befehl muß als Parameter zusätzlich die Länge der in der Tabelle abgelegten Ordinatenwerte (Byte, 16-bit-Wort, 32-bit-Doppelwort) angegeben werden. Die beiden zuerst genannten Befehls-Varianten runden das berechnete Ergebnis zur nächst gelegenen ganzen Zahl (*Round to Nearest Integer*). Die Länge des Ergebnisses entspricht dann der Länge der Ordinatenwerte.

Die beiden zuletzt genannten Befehls-Varianten liefern Ergebnisse mit einem 8-stelligen *Fractional*-Teil und einem Integer-Teil. Für Byte- und Wort-Ordinaten stimmt die Länge des Integer-Teils mit der Länge der Ordinatenwerte überein. Das Ergebnis wird hier durch das Vorzeichenbit auf 32 bit verlängert (*Sign Extension*). Dagegen wird für Doppelwort-Ergebnisse der Integer-Teil auf seine niederwertigen 24 Bits abgeschnitten, so daß er mit dem 8-bit-Fractional in ein 32-bit-Register paßt.

Eine einfachere Variante des TBL-Befehls erlaubt es nur, ein Tupelpaar (X1,Y1), (X2,Y2) in einem Registerpaar vorzugeben und – analog zu dem oben beschriebenen Verfahren – zu einem zwischen den Abszissenwerten X1 und X2 liegenden Wert X durch lineare Interpolation den entsprechenden Ordinatenwert Y zu bestimmen (*Register Interpolate Mode*).

[9] Dies entspricht einer Integer-Multiplikation mit X-1...X-8 und anschließendem Setzen des Punktes, so daß eine m.8-Zahl entsteht, d.h. das niederwertige Ergebnisbyte wird als *Fractional* interpretiert.