Mikrocomputertechnik mit dem Controller 68332

Schaltungstechnik
Maschinenorientierte Programmierung
Anwendungen

von
Professor Dipl. Ing. Günter Schmitt

mit 118 Bildern und 60 Programmbeispielen und
30 Übungsaufgaben

R. Oldenbourg Verlag München Wien 1998

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Schmitt, Günter:
Mikrocomputertechnik mit dem Controller 68332 : Schaltungstechnik, maschinenorientierte Programmierung, Anwendungen ; mit 60 Programmbeispielen und 30 Übungsaufgaben / von Günter Schmitt. - München ; Wien : Oldenbourg, 1998
ISBN 3-486-24622-4

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0, Internet: http://www.oldenbourg.de

Lektorat: Elmar Krammer
Herstellung: Rainer Hartl
Umschlagkonzeption: Kraxenberger Kommunikationshaus, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: Grafik + Druck, München
Bindung: R. Oldenbourg Graphische Betriebe GmbH, München

Inhaltsverzeichnis

Vorwort

Dieses Buch wendet sich an Studierende technischer Fachrichtungen, an industrielle Anwender und an technisch Interessierte, die den Einstieg in die faszinierende Welt der Computer suchen.

Die "Mikrocomputerwelt" entstand etwa ab 1975 mit den Universalprozessoren (8085, Z80, 6800 und 6502) und entwickelte sich in zwei Richtungen. Die eine ist die Welt der Datenverarbeitung mit ihren Personal Computern (PC) und die andere sind die technischen Anwendungen mit ihren vielfältigen Steuerungsaufgaben, die vom einfachen Fahrradtachometer bis zur elektronischen Vermittlungseinrichtung in der Telekommunikation reichen. Während jede neue Errungenschaft der PC Welt in der Öffentlichkeit Begeisterung und einen wahren Kaufrausch auslöst, sind die technischen Anwendungen der Mikrocomputer so selbstverständlich geworden, daß sie kaum noch beachtet werden. Der Werbespruch: "Wir haben in unsere elektronische Zahnbürste einen Computer eingebaut!" ruft nur noch ein müdes Lächeln hervor.

Wie sieht es mit der Aus- und Weiterbildung im Fach "Mikrocomputertechnik" aus? Was sollte gelehrt werden und was ist in 4 bis 6 Wochenstunden lehrbar?

Das sind sicher einmal die Grundlagen der digitalen Rechentechnik in Verbindung mit der Digitalelektronik, die Grundlagen der Programmierung in Verbindung mit einer höheren Programmiersprache und dann natürlich technische Anwendungen mit Beispielen und Übungen im Hinblick auf spätere berufspraktische Arbeitsfelder.

Unter diesen Gesichtspunkten scheiden die Superprozessoren der PC Welt als Beispiel für "den" Mikroprozessor und damit als Lehrgegenstand aus. Sie sind zu komplex, unzugänglich in Gehäuse eingebaut und einem sehr schnellen Wandel unterworfen. Die Schaltungsentwicklung ist auf einen kleinen Kreis von Spezialisten beschränkt, programmiert wird der PC, wenn überhaupt noch, in höheren Sprachen, und die orientieren sich mehr an abstrakten Modellen als an realer Technik.

Daher haben sich nach dem Aussterben der erwähnten Universalprozessoren, die anfangs als Lehrgegenstand dienten, die Mikrocontroller in der Lehre durchgesetzt. Die Auswahl fällt schwer. Hier 8 bit, dort 16 und 32 bit. Hier Intel mit seinen Abkömmlingen, dort Motorola mit seinen Anhängern und nun auch noch die PIC- und ST6-Familien. Der Preis ist in der Ausbildung sicher nicht das entscheidende, sondern eher didaktische Gesichtspunkte. Dazu gehören einfache und klare Register- und Befehlssätze sowie übersichtliche Speicherstrukturen in Verbindung mit vielfältigen Adressierungsarten.

Dieses Buch behandelt den Controller 68332 aus der 68300-Familie des Herstellers Motorola. Mit seiner 32 bit Struktur, der umfangreichen Peripherie und einem Adressierungsbereich von 16 MByte gehört er zur oberen Leistungsklasse. Vom Register- und Befehlssatz sowie von seiner Speicherstruktur her gesehen erfüllt er alle didaktischen Anforderungen. Schwieriger ist die Handhabung der komplexen seriellen Peripherie und der Timereinheit. Hier wird man für den Einstieg eine Unterprogrammbibliothek für die wichtigsten Anwendungsfälle bereithalten müssen.

Kapitel 1 gibt eine Einführung in die Funktionseinheiten eines Mikrocomputers und erklärt dann die Unterschiede zwischen dem Personal Computer und den Controlleranwendungen. Kapitel 2 behandelt die Hardware des 68332 unter besonderer Berücksichtigung der Möglichkeit, durch Programmierung von Steuerregistern den Controller an eine Vielzahl von Anwendungen anzupassen. Arbeitet man mit einer vorgegebenen Schaltung, so sollte man dieses Kapitel nur als zusätzliche Information betrachten. Kapitel 3 behandelt den Register- und Befehlssatz des 68332 mit vielen lauffähigen Beispielprogrammen und 30 Übungsaufgaben, für die im Kapitel 6 Lösungsvorschläge bereitgestellt werden. Dabei werden Testdaten sowohl über die digitalen Schnittstellen als auch auf einem PC als Terminal ein- und ausgegeben. Kapitel 5 enthält eine alphabetisch geordnete Befehlsliste. Die Anwendungsbeispiele des Kapitels 4 behandeln die Systemtimer, die seriellen Schnittstellen, analoge und digitale Peripherie sowie die Timereinheit mit ihren festen Timerfunktionen.

Zu diesem Buch ist eine Diskette mit den Programmbeispielen, den Lösungen der Übungsaufgaben und den Unterprogrammen des Anhangs sowie einem Cross-Assembler erhältlich, der unter DOS bzw. im DOS-Fenster von Windows 3.11 und Windows 95 läuft. Mit der Bestellkarte auf der letzten Seite können auch Informationen zum Bezug der Hardware angefordert werden, die auf das Buch zugeschnitten ist.

Ich danke meiner Frau für die Hilfe bei der Korrektur und die moralische Unterstützung sowie Herrn Dr. Weiß, Schriesheim, für die Anregung zu diesem Buch und die Hilfestellung bei der Herstellung der Hardware.

Günter Schmitt

1. Einführung

Dieses Kapitel beschreibt die *Funktionseinheiten* eines Mikrocomputers. Darunter sollen die *Komponenten* verstanden werden, die für den Betrieb eines Rechners erforderlich sind. Eilige Leser, die schon Erfahrungen mit Mikroprozessoren haben, können dieses Kapitel überschlagen.

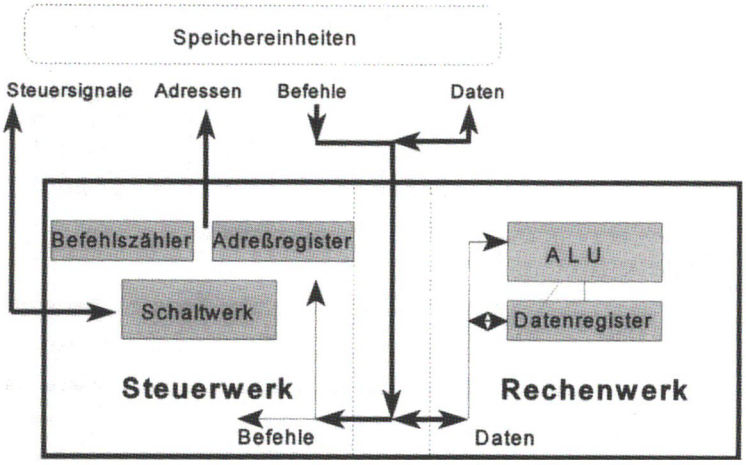

Bild 1-1: Der Aufbau eines Mikroprozessors

Der in *Bild 1-1* dargestellte ***Mikroprozessor*** wird auch als Zentraleinheit (CPU Central Processing Unit) bezeichnet. Parallele Leitungsbündel, auch Bus genannt, verbinden den Prozessor mit den Speicher- und Peripherieeinheiten. In der von-Neumann-Struktur liegen die Befehle und Daten in einem gemeinsamen Speicherbereich und werden durch *Adressen* ausgewählt. *Maschinenbefehle* sind binär codierte Anweisungen an das Steuerwerk, bestimmte Operationen auszuführen; *Daten* sind die zu bearbeitenden Zahlen, Zeichen oder andere binär codierte Größen.

Das ***Rechenwerk*** enthält Schaltungen zum Berechnen und Speichern der Daten. Die arithmetisch-logische Einheit (ALU Arithmetic Logical Unit) ist ein Schaltnetz, das arithmetische Operationen in den vier Grundrechenarten und logische Operationen wie z.B. UND-Verknüpfungen bitparallel durchführt. Die Operanden werden kurzzeitig in *Datenregistern* zwischengespeichert und über den Datenbus zwischen dem Prozessor und dem Datenspeicher übertragen.

Das ***Steuerwerk*** enthält ein Schaltwerk, das die Befehle in Steuersignale umsetzt. Bei einem Mikroprogrammsteuerwerk liegen die Verarbeitungsschritte in Form von Mikrobefehlen in einem prozessorinternen Festwertspeicher; der Mikrocode

dieses "Computers auf dem Prozessor" ist dem Anwender nicht zugänglich. Der *Befehlszähler* enthält immer die Adresse des nächsten Maschinenbefehls, der aus dem Befehlsspeicher über den Datenbus in das Steuerwerk geholt wird. *Adreßregister* dienen zur Speicherung und Berechnung von Adressen. Das Steuerwerk legt entweder eine Befehls- oder eine Datenadresse auf den Adreßbus; die Übertragung der adressierten Operanden über den gemeinsamen Bus, Datenbus genannt, wird durch Signale gesteuert. Befehle gelangen in das Steuerwerk, Daten in das Rechenwerk. In der Harvard-Struktur aufgebaute Rechner können durch getrennte Bussysteme gleichzeitig Befehle und Daten übertragen.

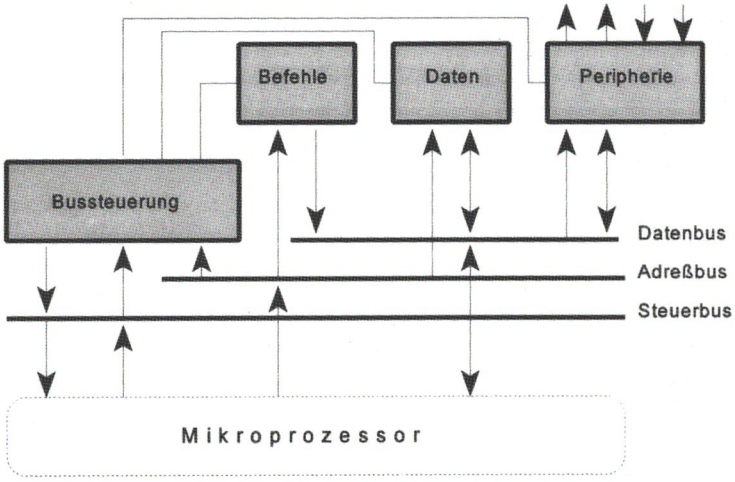

Bild 1-2: Die Speicher- und Peripherie-Einheiten

Die Speicher- und Peripherie-Einheiten werden entsprechend *Bild 1-2* an den Adreß-, Daten- und Steuerbus des Mikroprozessors angeschlossen. Ein Decoder der ***Bussteuerung*** wählt mit den höherwertigen Adreßleitungen eine der angeschlossenen Einheiten aus, Steuersignale des Prozessors bestimmen die Richtung (Lesen bzw. Schreiben) und den Zeitpunkt der Übertragung. Die niederwertigen Adreßleitungen adressieren über einen Decoder, der sich auf der ausgewählten Einheit befindet, die entsprechende Speicherstelle. Die Bussteuerung liefert dem Mikroprozessor Bestätigungssignale, wenn die adressierte Einheit die Datenübertragung beendet hat.

In einem *Lesezyklus* legt der Prozessor die Adresse der Speicherstelle auf den Adreßbus. Die ausgewählte Einheit schaltet den Inhalt (Befehl oder Datum) auf den Datenbus; alle anderen Einheiten sind abgeschaltet (tristate). Der Prozessor übernimmt die Daten vom Datenbus, gesteuert durch das Bestätigungssignal der Bussteuerung.

In einem *Schreibzyklus* legt der Prozessor die Adresse der Speicherstelle auf den Adreßbus und die Daten auf den Datenbus. Die Bussteuerung gibt die entsprechende Einheit frei, die nun die Daten übernimmt. Mit dem Bestätigungssignal der Bussteuerung entfernt der Prozessor die Daten vom Datenbus.

Bei ***Speichereinheiten*** unterscheidet man nichtflüchtige Festwertspeicher und flüchtige Schreib-Lese-Speicher. Ein *Festwertspeicher* hat beim Einschalten der Versorgungsspannung einen vorgegebenen Speicherinhalt, der im Betrieb nicht geändert werden kann und der beim Abschalten der Versorgungsspannung erhalten bleibt. Die Bauform ROM (Read Only Memory) wird beim Hersteller maskenprogrammiert. In der Bauform EPROM (Erasable Programmable ROM) programmiert der Anwender den Baustein in besonderen Geräten und kann ihn durch Bestrahlen mit UV-Licht wieder löschen. In der Bauform EEPROM (Electrical Erasable Programmable ROM) bzw. EAROM (Electrical Alterable ROM) kann der Speicherinhalt auch während des Betriebes durch besondere Verfahren geändert werden. *Schreib-Lese-Speicher* werden auch als RAM (Random Access Memory) bezeichnet. Beim Einschalten der Versorgungsspannung haben sie einen zufälligen Speicherinhalt. Während des Betriebes werden sie beschrieben und gelesen. Der Speicherinhalt geht nach dem Abschalten der Versorgungsspannung verloren; in der Bauform BRAM (Batteriegepuffertes RAM) bleibt er erhalten. Dynamische RAM Bausteine (DRAM) müssen durch besondere Signale periodisch aufgefrischt werden; statische RAM Bausteine (SRAM) benötigen dies nicht.

Speicherbausteine sind in der Regel byteorganisiert, d.h. durch 8 parallele Datenleitungen werden immer 8 Bits gelesen oder beschrieben. Für besondere Anwendungen gibt es wortorganisierte Bausteine mit 16 parallelen Datenanschlüssen und bitorganisierte EEPROM Bausteine, die seriell gelesen und beschrieben werden.

Die Einteilung der Mikroprozessoren und Mikrocomputer nach ihrer *Bitbreite* ist nicht immer eindeutig. Darunter versteht man entweder die Größe des Datenbus, also die Anzahl der parallelen Datenleitungen oder die Größe der Datenregister im Prozessor. Die Controller der 683xx Familie enthalten 32 bit breite Daten- und Adreßregister, der Datenbus kann sowohl mit 8 als auch mit 16 bit betrieben werden. Die meisten Befehle lassen sich wahlweise auf 8 bit oder 16 bit oder 32 bit breite Daten anwenden.

Die Größe des Adreßbus, also die Anzahl der Adreßleitungen, bestimmt den maximal adressierbaren Speicherbereich. Mit 24 Adreßbits lassen sich 16 MByte adressieren, die nur in wenigen Anwendungen voll ausgenutzt werden. Die Prozessoren der 683xx Familie haben keine besonderen Peripheriebefehle und Peripheriesignale; die Peripheriebausteine liegen mit den Speicherbausteinen in einem gemeinsamen Speicherbereich.

Als ***Peripherie*** bezeichnet man die an den Rechner angeschlossenen Geräte wie z.B. Drucker oder Tastaturen, die über Peripherie- oder Interfaceeinheiten am Bus betrieben werden. *Bild 1-3* zeigt die digitale Signalübertragung.

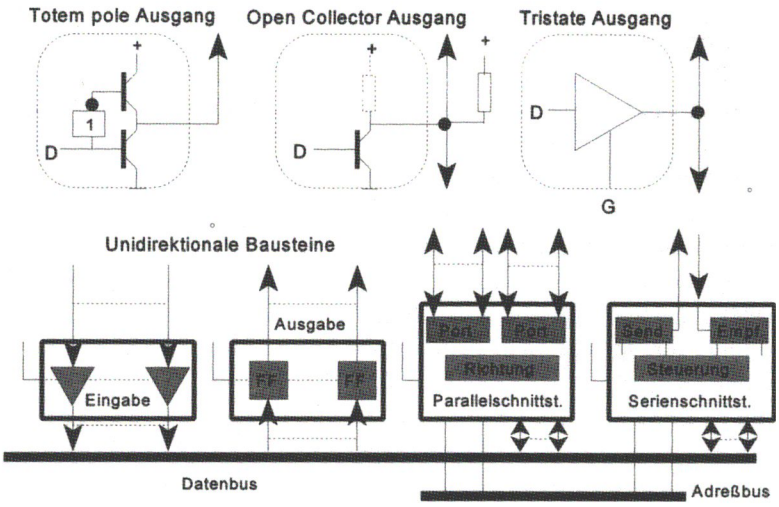

Bild 1-3: Die digitale Peripherie

Die *digitalen Peripherieeinheiten* liegen am Datenbus des Prozessors und arbeiten peripherieseitig vorwiegend mit TTL-Pegel. Parallele Schnittstellen sind meist byteorganisiert; mit Bitbefehlen lassen sich jedoch auch einzelne Bitpositionen innerhalb des Bytes eingeben (lesen) oder ausgeben (schreiben).

Die *auszugebenden Daten* werden in Flipflops gespeichert. Man unterscheidet drei Ausgangstreiberschaltungen. Ein *Totem pole Ausgang* gibt immer ein festes Potential (High oder Low) aus und läßt sich daher nicht mit anderen Ausgängen parallel schalten. Bei einem *Open Collector* bzw. *Open Drain Ausgang* hält ein interner oder externer Widerstand das Potential auf High, das nun entweder mit dem Ausgangstransistor oder einer anderen parallel liegenden Schaltung auf Low gezogen werden kann. Der *Tristate Ausgang* läßt sich mit einem Steuereingang hochohmig machen und damit abschalten; dies ist die bevorzugte Ausgangsschaltung für Bussysteme wie z.B. den Datenbus.

Bei der *Eingabe* von digitalen Signalen wird das Potential der Peripherieleitung meist über einen Tristate Treiber auf den Datenbus geschaltet und nicht in der Peripherieeinheit gespeichert. *Bidirektionale* Einheiten lassen sich sowohl zur Eingabe als auch zur Ausgabe von Signalen verwenden. Dazu sind Open Collector bzw. Open Drain oder Tristate Ausgänge erforderlich. Bei programmierbaren Schnittstellen wird die Richtung (Eingabe oder Ausgabe) in Richtungsregistern festgelegt.

Serielle Schnittstellen dienen zur Umwandlung des parallelen Datenformats des Prozessors in einen seriellen Bitstrom und umgekehrt. Die Umsetzung geschieht durch Schieberegister. Beispiele sind der Sender bzw. Empfänger einer Serienschnittstelle zum Anschluß eines Modems oder einer Maus. Besondere Funktionen wie z.B. die Übertragungsgeschwindigkeit und Rahmenbits sind in Steuerregistern programmierbar.

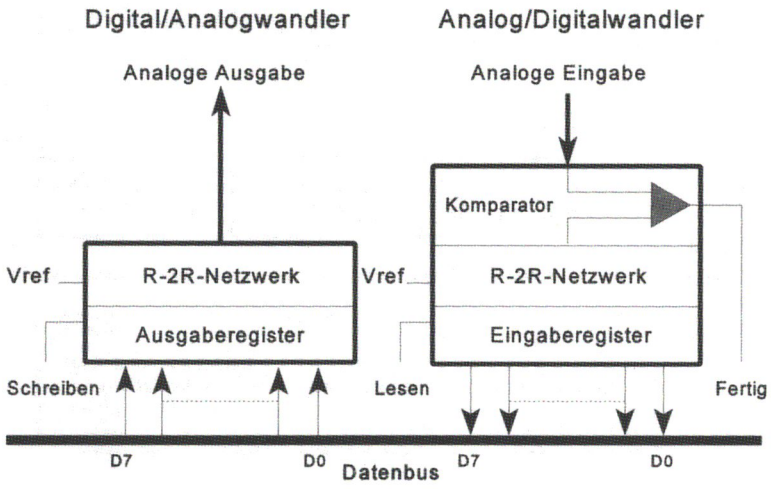

Bild 1-4: Die analoge Peripherie

Die in *Bild 1-4* dargestellten *analogen Peripherieeinheiten* sind an den parallelen Datenbus angeschlossen. Ein *Digital/Analogwandler* setzt einen binären Eingangswert in eine analoge Ausgangsspannung um. Der Wert 0 liefert z.B. eine Ausgangsspannung von 0 Volt, der Wert 255 eine Spannung von +5 Volt. Dazwischen liegen bei einem 8 bit Wandler noch 253 weitere Spannungsstufen. Die auszugebenden Daten werden in ein Ausgaberegister geschrieben. Ein Netzwerk bewertet die Bitpositionen und summiert sie über eine Referenzspannung V_{ref}. Die Umsetzzeit zwischen dem Einschreiben des digitalen Wertes und der analogen Ausgabe hängt nur von der Schaltzeit der Bauteile ab und liegt bei etwa 1 µs. Der in Bild 1-4 dargestellte *Analog/Digitalwandler* vergleicht die zu messende Eingangsspannung mit einer von einem Digital/Analogwandler erzeugten Vergleichsspannung. Die Umsetzzeit liegt beim Rampenverfahren bei ca. 10 ms, beim Verfahren der schrittweisenden Näherung bei ca. 10 µs und bei Parallelumsetzern unter 500 ns. Das Ende der Umsetzung wird durch ein Fertigsignal gemeldet. Neben den dargestellten parallelen Wandlern gibt es serielle Wandler, die einen digitalen seriellen Bitstrom in einen analogen Wert bzw. einen analogen Wert in einen digitalen seriellen Bitstrom umsetzen.

Ein *Timer* ist eine Funktionseinheit, die Zählregister zum Einstellen von Verzögerungszeiten, zur Erzeugung von periodischen Uhrenimpulsen und zum Zählen von Flanken enthält.

Aus der Vielzahl der Bauformen und Anwendungen sollen zwei charakteristische Beispiele herausgegriffen werden, ein PC (Personal Computer) als Datenverarbeitungsanlage und ein Mikrocontroller zur Gerätesteuerung.

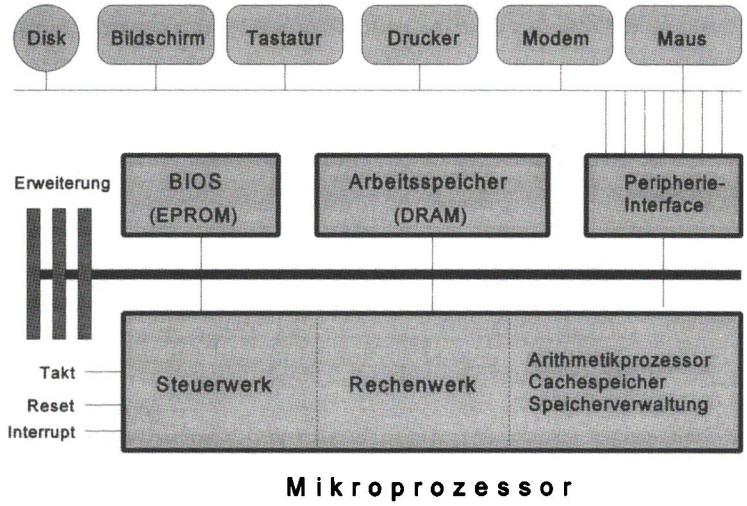

Bild 1-5: Der Aufbau eines Personal Computers

Die in *Bild 1-5* dargestellte Schaltung eines Personal Computers enthält einen für die besonderen Aufgaben der ***Datenverarbeitung*** entworfenen *Mikroprozessor*, der neben dem Steuerwerk und Rechenwerk einen Arithmetikprozessor (Gleitpunktbefehle), einen Schnellzugriffsspeicher (Cache) und eine Speicherverwaltungseinheit (MMU Memory Management Unit) enthält. Beim Einschalten des Gerätes bzw. bei einem Reset wird ein Basissystem (BIOS Basic Input Output System) aus einem *Festwertspeicher* (EPROM) gestartet, das ein Betriebssystem (DOS, Windows, UNIX) von der Festplatte in den *Arbeitsspeicher* lädt. Dieser ist aus dynamischen Speichern aufgebaut und nimmt neben dem Betriebssystem weitere Systemprogramme (Textverarbeitung) und Benutzerprogramme (C oder Pascal) sowie Daten (Texte) auf. Die *Peripherieeinheiten* Disk (Festplatte und Floppy) sowie der Bildschirm (Video) werden meist über Interfacekarten an Buserweiterungssteckplätzen betrieben. Für die Tastatur und den Drucker sowie für die seriellen Einheiten Maus und Modem sind die Interfacebausteine direkt an den Bus angeschlossen.

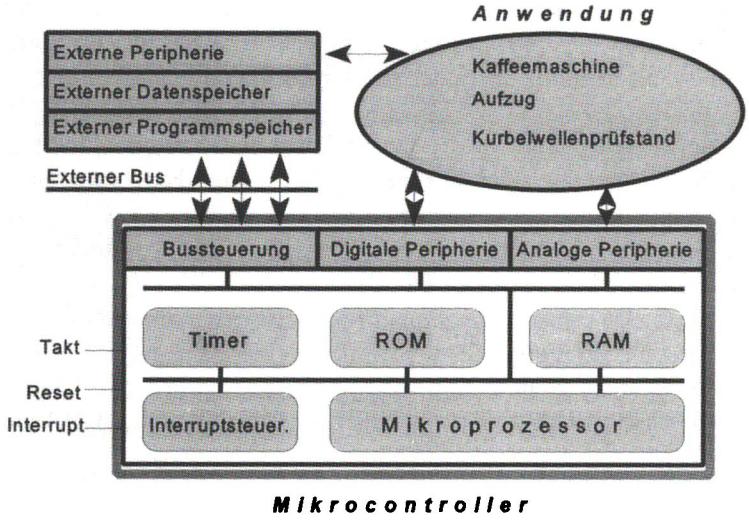

Bild 1-6: Der Aufbau eines Controllers zur Gerätesteuerung

Der in *Bild 1-6* dargestellte ***Mikrocontroller*** dient je nach Ausführung zur Steuerung von Kleingeräten (Kaffemaschine, Waschmaschine) oder größeren Anlagen (Kurbelwellenprüfstand, Gummistiefelpresse). Er enthält alle Funktionseinheiten eines Computers auf einem Baustein (Chip). Dazu zählen der Mikroprozessor, ein Festwertspeicher (ROM) für das Programm und konstante Daten sowie ein Schreib-Lese-Speicher (RAM) für Variablen. Die Peripherieeinheiten Timer sowie parallele, serielle und analoge Schnittstellen sind auf dem Baustein integriert und werden zur Anwendungsschaltung herausgeführt. Je nach Bauform und Aufgabe können zusätzliche Speicher- und Peripheriebausteine an einen externen Bus angeschlossen werden.

Mikrocontroller sind Rechnereinheiten, die für die besonderen Aufgaben der industriellen Steuerung entwickelt wurden, um die alten 8 bit Mikroprozesoren (8085, Z80 und 6800) abzulösen. Sie werden auch als Embedded Controller bezeichnet. Kennzeichen:
- Integration von Prozessor, Speicher und Peripherie auf einem Baustein,
- Anschluß zusätzlicher Speicher- und Peripherieeinheiten möglich,
- programmierbare Funktionen für Bausteinauswahl und Peripheriesteuerung,
- dadurch keine oder nur wenige zusätzliche externe Hilfsbausteine,
- niedrige Bausteinkosten durch hohe Integrationsdichte und Massenfertigung,
- wirtschaftliche Herstellung von Geräten mit einem einzigen Steuerbaustein,
- Programmierung sowohl in Maschinensprache als auch in Hochsprachen und
- Gerätefunktionen durch Programmänderung erweiterbar.

Die Einsatzgebiete der Mikrocontroller reichen von einfachen Gerätesteuerungen (z.B. elektronische Zahnbürste) bis hin zu aufwendigen Anwendungen in der Telekommunikation (z.B. Netzwerkrechner). Aus der Vielzahl von Bauformen und Herstellern seien einige charakteristische Familien herausgegriffen:

Als Intel-Familie bezeichnet man die Typen 80xxx des Herstellers Intel, die von vielen anderen Firmen zunächst in Lizenz gefertigt und dann weiterentwickelt wurden. Dazu gehören die Mikrocontroller 8051, 80535 und 80C166. Der Typ 80535 des Herstellers Siemens als Beispiel ist gekennzeichnet durch:
- interne und externe Datenstruktur 8 bit,
- Version ohne ROM erfordert externen Programmspeicherbaustein,
- externer Adreßbus 16 bit (64 kByte externer Speicher),
- interner RAM 256 Bytes und
- interne Peripherie (4 Parallelports, 3 Timer, V.24-Schittstelle, A/D-Wandler).

Als Motorola-Familie bezeichnet man die Typen 68xxx des Herstellers Motorola und anderer. Dazu gehören die Typen 68HC05, 68HC11 und die 683xx-Familie. Der Typ 68HC11 entspricht in seiner 8 bit Struktur im Wesentlichen dem Typ 80535 der Intel-Familie. Die 683xx-Familie ist gekennzeichnet durch einen modularen Aufbau, der aus einem CPU-Kern der Datenbreite 32 bit und der Adreßbreite von 32 bit sowie einer Vielzahl von besonderen Funktionseinheiten besteht, die den Baustein dem jeweiligen Anwendungsgebiet anpassen. Dazu gehören u.A. die Typen:
- 68302 mit besonderem Kommunikationsprozessor (DMA-Controller),
- 68331 mit besonderem GPT (General Purpose Timer),
- **68332** mit besonderer TPU (Time Processor Unit) und
- 68360 mit erweitertem Kommunikationsmodul.

Neuere Erscheinungen sind die PIC-Familie des Herstellers Microchip und die ST6-Familie des Herstellers SGS-Thompson.

2. Die Hardware des 68332

Der 8 bit Prozessor 6800 ist der Urvater der Prozessorfamilien des Herstellers Motorola; die 8 bit Peripheriebausteine wie z.B. die Parallelschnittstelle 6821 lassen sich auch für 32 bit Prozessoren verwenden. Der nachfolgende Prozessor 68000 mit einer 32 bit internen Datenstruktur und einem 16 bit breiten Datenbus wurde in der Datenverarbeitung und für Steuerungsaufgaben eingesetzt. Aus ihm entstanden sowohl die 32 bit Prozessoren 680xx für den Einsatz in Personal Computern als auch die Controller mit den Bezeichnungen 683xx wie z.B. der hier behandelte 68332 für Steuerungen. Sie werden auch als Embedded (eingebettete) Controller oder MCU (Micro Controller Unit) bezeichnet und enthalten als Kern die CPU32, einen weiterentwickelten 68000 Prozessor. Je nach Einsatzgebiet sind auf den Bausteinen parallele und serielle Schnittstellen, Timer, Arbeitsspeicher (RAM) und besondere Steuerschaltungen integriert.

2.1 Die Funktionsblöcke und Steuerregister

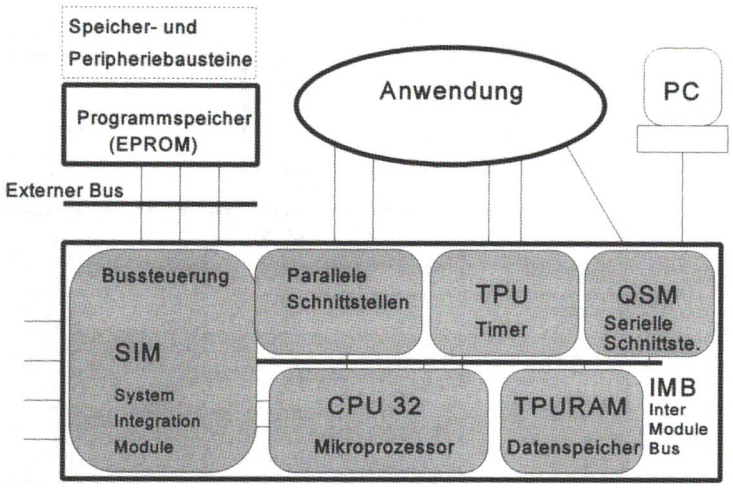

Bild 2-1: Die Moduleinheiten des 68332

Die Funktionseinheiten oder Moduln des 68332 (*Bild 2-1*) liegen an einem internen Bus, dem IMB (Inter Module Bus). Ein System Integration Module (SIM) dient zur Einstellung der Betriebsarten des Bausteins. Da kein interner Programmspeicher vorgesehen ist, muß über eine Bussteuerung mindestens ein externer Festwertspeicher (EPROM oder EEPROM) angeschlossen werden. Die Prozessoreinheit CPU 32 arbeitet intern mit 32 bit Registern, auf dem Datenbus

aber nur mit 16 bit. Der interne Schreib-Lese-Speicher (RAM) wird vorwiegend als Datenspeicher benutzt, kann aber auch als TPURAM besondere Mikroprogramme des Timers (TPU Time Processor Unit) aufnehmen. An eine der seriellen Schnittstellen (QSM Queued Serial Module) wird in der Entwicklungs- und Testphase ein Terminal bzw. ein PC mit Entwicklungssoftware angeschlossen. Ein Teil der Anschlüsse des Controllers läßt sich entweder als Busleitung oder als Steuerleitung oder als Parallelportanschluß verwenden. Für die Programmierung der Modulfunktionen stehen ca. 250 Steuerregister zur Verfügung, deren Anwendung in den folgenden Abschnitten erläutert wird. Sie liegen nach dem Einschalten im oberen Adreßbereich und lassen sich nach Bedarf in einen anderen Bereich verlegen. Die Bezeichnungen der Register und Bitpositionen in den Tabellen und Beispielen entsprechen denen des Herstellersystemhandbuchs.

2.2 Die Reset- und Taktsteuerung

Bild 2-2 gibt einen Überblick über die Verhältnisse beim Einschalten bzw. bei einem Reset durch ein von außen angelegtes Signal.

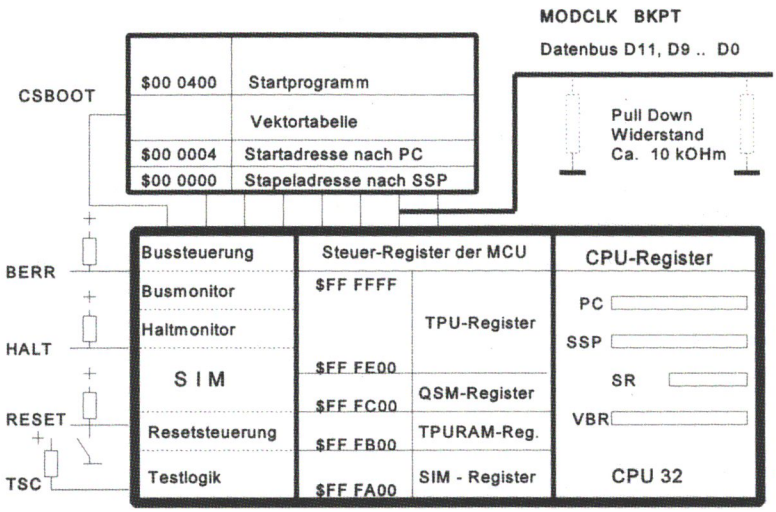

Bild 2-2: Der Anlauf des Systems mit Reset

Die Steuereingänge BERR (Busfehler), HALT (Haltsteuerung), RESET (Zurücksetzen) und TSC (Tristatesteuerung) müssen beim Anlauf mit Widerständen von 3,3 bis 10 kOhm auf High gehalten werden. Äußere Schaltungen für HALT, BERR und RESET müssen Open Collector bzw. Open Drain Ausgänge haben, da der Controller diese Anschlüsse auch als Ausgänge benutzt. Beim Anlauf des Systems werden die Steuerleitungen MODCLK (Taktbetriebsart) und BKPT (Hintergrundbetrieb) sowie die Datenbusleitungen D0 bis D9 und D11 von der

Systemsteuerung durch interne Pull Up Widerstände auf High gehalten. Mit externen Pull Down Widerständen, die nur in der Anlaufphase wirksam sind, kann der Benutzer entsprechend *Bild 2-3* einen Teil der Betriebsarten des Controllers voreinstellen. Die mit den Potentialen dieser Leitungen vorbesetzten Bitpositionen der Steuerregister lassen sich während des Betriebes durch Befehle ändern.

Leitung	**High** (offen lassen)	**Low** (durch Widerstand)
Datenbus D0	16 bit Datenbus für CSBOOT	8 bit Datenbus für CSBOOT
Datenbus D1	Bausteinauswahl CS0, CS1, CS2	Steuersignale BR, BG, BGACK
Datenbus D2	Bausteinauswahl CS3, CS4, CS5	Statussignale FC0, FC1, FC2
Datenbus D3	Bausteinauswahl CS6	Adreßleitung A19
Datenbus D4	Bausteinauswahl CS6 bis CS7	Adreßleitungen A19 bis A20
Datenbus D5	Bausteinauswahl CS6 bis CS8	Adreßleitungen A19 bis A21
Datenbus D6	Bausteinauswahl CS6 bis CS9	Adreßleitungen A19 bis A22
Datenbus D7	Bausteinauswahl CS6 bis CS10	Adreßleitungen A19 bis A23
Datenbus D8	Ausgabe von Bussteuersignalen	Parallelschnittstelle Port E
Datenbus D9	Eingänge für Interruptsignale	Parallelschnittstelle Port F
Datenbus D11	Testbetriebsart gesperrt	Testbetriebsart freigegeben
Eingang MODCLK	Interner Taktgen., Vorteiler = 1	Externer Taktgen., Vort. = 512
Eingang BKPT	Hintergrundbetriebsart gesperrt	Hintergrundbetriebsart frei

Bild 2-3: Die Voreinstellungen bei Reset

In dem Minimalsystem des Abschnitts 2.8 wurden mit Pull Down Widerständen folgende Voreinstellungen gewählt:
- D0 auf Low gelegt: Anlauf als 8 bit System mit dem Steuersignal CSBOOT,
- D8 auf Low gelegt: keine Ausgabe von externen Bussteuersignalen und
- D9 auf Low gelegt: keine externen Interrupts, sondern parallele Schnittstelle.

Alle anderen Anlaufsteuerleitungen wurden nicht beschaltet, sondern blieben High und ergaben die Voreinstellungen:
- Bausteinauswahlsignale anstelle von Steuer- und Adreßbussignalen,
- kein Testbetrieb,
- interner Taktgenerator und Vorteiler für Systemtimer = 1 sowie
- keine Hintergrundbetriebsart.

Beim Einschalten der Versorgungsspannung (Power On Reset = Einschaltreset) legt die interne Resetsteuerung die RESET Leitung für die notwendige Anlauf-

zeit auf Low; externe Schaltungen sollten den Eingang ca. 100 ms auf Low halten und müssen Open Collector bzw. Open Drain Ausgänge haben. Bei einem Reset während des Betriebes (Push Button Reset = Tastenreset) sollte das Signal mindestens 10 Takte anliegen. In einfachen Anwendungen genügt ein - nicht entprellter - Taster gegen Low-Potential.

Mit der steigenden Flanke des Reset Signals startet das System. Die Steuerregister der Moduleinheiten sind mit Anfangswerten vorbesetzt, ihre Adressen liegen zunächst im oberen Adreßbereich von $FF FA00 bis $FF FFFF. In der CPU wird das Vektorbasisregister VBR gelöscht, die Vektortabelle liegt im unteren Adreßbereich von $00 0000 bis $00 03FF. Das Statusregister SR erhält folgende Voreinstellungen:
- Statusbit S = 1: Systemstatus,
- Tracebits T0 = T1 = 0: keine Einzelschrittsteuerung und
- I0 = I1 = I2 = 1: alle Interrupts mit Ausnahme von NMI gesperrt.

Die Bussteuerung liest mit dem Signal CSBOOT (Bausteinauswahl beim Systemstart) das auf der Adresse $00 0000 liegende Langwort und lädt es in den Systemstapelzeiger SSP = A7 für S = 1. Der Befehlszähler wird mit dem nächsten Langwort von der Adresse $00 0004 geladen; dies ist die Startadresse des nun auszuführenden Programms. Das folgende Beispiel legt den Stapel auf die Adresse unterhalb $10 0800 und startet ein Programm ab Adresse $00 0400.

```
stapel  EQU     $100800     ; Stapel im oberen TPURAM
        ORG     $0          ; BOOT-Vektoren
        DC.L    stapel      ; nach Stapelzeiger SSP
        DC.L    start       ; nach Befehlszähler PC
; Einsprünge der Vektortabelle nicht dargestellt
        ORG     $400        ; Anfang des Startprogramms
start   MOVE.B  #0,SYPCR    ; 1. Befehl
```

Das Startprogramm muß nun in den Modulsteuerregistern die gewünschten Betriebsarten einstellen. Alle Bitpositionen mit Ausnahme der durch U gekennzeichneten Anzeigebits sind bei Reset mit 0 oder 1 bzw. über die Pull Down Widerstände vorbesetzt. Die folgenden Beispiele beschreiben nur die Betriebsarten des Minimalsystems; für weiterführende Aufgaben sollte das Systemhandbuch des Herstellers herangezogen werden, dessen Bezeichnungen beibehalten wurden. Wird der Adreßbereich der Register nach $7F Fxxx verlagert, so sind die Adressen entsprechend zu ändern. Die Register lassen sich mit Wort- und mit Bytebefehlen adressieren. Beim Bytezugriff liegt das High-Byte des Wortes auf der Wortadresse und das Low-Byte des Wortes auf der folgenden. Beispiele:

```
MOVE.W  #$1234,$FFFA00 ; Wortzugriff nach SIMCR
MOVE.B  #$12,$FFFA00   ; Byte nach High-Teil von SIMCR
MOVE.B  #$34,$FFFA01   ; Byte nach Low-Teil von SIMCR
MOVE.W  #$1234,SIMCR   ; symbolische Adressierung
MOVE.B  #$12,SIMCR     ; SIMCR vordefiniert
MOVE.B  #$34,SIMCR+1   ; Adreßberechnung Assembler
```

Das SIM Configuration Register **SIMCR** auf der Wortadresse $FF FA00 legt die Arbeitsweise des Systemsteuermoduls (SIM) fest. Zugriff nur im Systemstatus.

High-Byte (Byteadresse $FF FA00)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
EXOFF	FRZSW	FRZBM	0	SLVEN	0	SHEN	SHEN
Reset:0	0	0	0	D11	0	0	0

Low-Byte (Byteadresse $FF FA01)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUPV	MM	0	0	IARB	IARB	IARB	IARB
Reset:1	1	0	0	1	1	1	1

Die Bitpositionen haben folgende Bedeutung:
EXOFF = 0: Taktausgang CLKOUT gibt internen Systemtakt aus
EXOFF = 1: Taktausgang CLKOUT ist tristate

FRZSW = 0: Watchdog und Systemtimer im Testbetrieb frei
FRZSW = 1: Watchdog und Systemtimer im Testbetrieb gesperrt

FRZBM = 0: Busmonitor im Testbetrieb frei
FRZBM = 1: Busmonitor im Testbetrieb gesperrt

SLVEN = 0: kein Zugriff auf internen Bus im Testbetrieb
SLVEN = 1: Zugriff auf internen Bus im Testbetrieb zugelassen

SHEN = 0 0: interner Bus nicht angezeigt, Busverfolgung möglich
SHEN = 0 1: interner Bus extern angezeigt, keine Busverfolgung
SHEN = 1 0: interner Bus extern angezeigt, Busverfolgung möglich
SHEN = 1 1: interner Bus extern angezeigt, Halt bei Busvergabe

SUPV = 0: Steuerregister im Benutzer- und Systemstatus zugänglich
SUPV = 1: Steuerregister nur im Systemstatus zugänglich (***Vorgabe***)

MM = 0: Modulsteuerregister ab Adresse $7F Fxxx
MM = 1: Modulsteuerregister ab Adresse $FF Fxxx (***Vorgabe***)
Das Bit MM ist nach einem RESET nur ***einma***l programmierbar!

IARB = xxxx: legt Interruptpriorität des SIM Moduls fest
IARB = 0000: keine Interrupts durch SIM Modul (Interruptfehler möglich!)
IARB = 0001: SIM Modul hat niedrigste Modulpriorität
IARB = 1111: SIM Modul hat höchste Modulpriorität (***Vorgabe***)

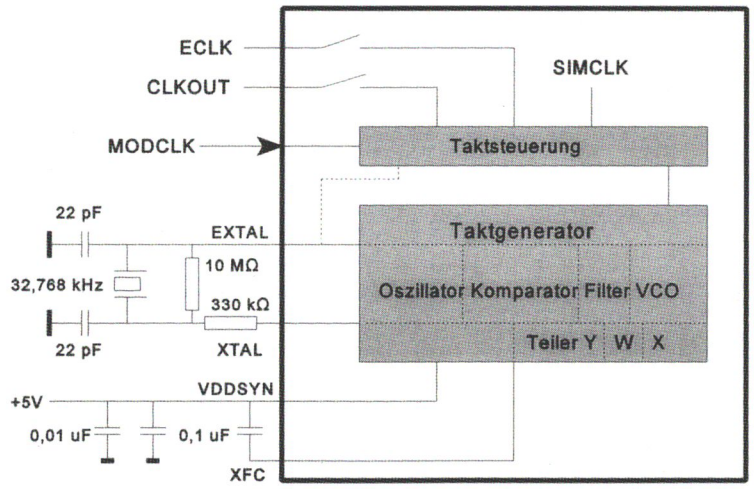

Bild 2-4: Der Taktgenerator

Der Eingang MODCLK des Taktgenerators (*Bild 2-4*) bestimmt beim Reset die Taktquelle. Wird der Eingang mit einem Pull Down Widerstand auf Low gehalten, so ist ein externer Takt an EXTAL einzuspeisen; der interne Taktgenerator ist abgeschaltet.

Liegt der Eingang MODCLK (unbeschaltet) beim Reset auf High, so erzeugt der interne programmierbare Taktgenerator den Systemtakt aus einem an EXTAL und XTAL angeschlossenen Quarz oder aus einem Referenztakt, der nur an EXTAL angelegt wird. Notfalls erzeugt der Taktgenerator aus einem internen RC-Glied einen eigenen Takt von ca. 8,4 MHz. Die Taktausgänge CLKOUT und ECLK sind abschaltbar.

Die Frequenz F_{Sys}des Systemtaktes berechnet sich aus der Frequenz F_{Ref} des Referenztaktes und den Teilern Y, W und X der Frequenzsteuerung, die im Register SYNCR programmiert werden, nach der Formel:

$$F_{Sys} = F_{Ref} * 4 * (Y + 1) * 2^{2*W + X}$$

Vorgabe nach Reset: W = 0, X = 0 und Y = $3F = 63: F_{sys} = 8,4 MHz.
Für einen Quarz von 32,768 kHz und dem Faktor Y = %111111 = 63 sowie den Teilern W = 0 und X = 1 ergibt sich durch Programmieren ein Systemtakt von 16,777 MHz.

Die Versorgungsspannung VDDSYN des Tiefpassfilters verlangt einen besonders verlustarmen Kondensator von 0,1 µF gegen den Eingang XFC, die anderen Kondensatoren glätten die Spannung.

Das Clock Synthesizer Control Register **SYNCR** auf der Wortadresse $FF FA04 bestimmt die Arbeitsweise des Taktgenerators. Zugriff nur im Systemstatus.

High-Byte (Byteadresse $FF FA04)

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
W	X	Y	Y	Y	Y	Y	Y
Reset:0	0	1	1	1	1	1	1

Low-Byte (Byteadresse $FF FA05)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EDIV	0	0	SLIMP	SLOCK	RSTEN	STSIM	STEXT
Reset:0	0	0	U	U	0	0	0

Die Bitpositionen haben folgende Bedeutung:

W = Teiler des Frequenzgenerators (VCO) entsprechend Formel F_{Sys} =

X = Vorteiler des Frequenzgenerators entsprechend Formel F_{Sys} =

Y = Zähler (0 bis 63) des Frequenzgenerators entsprechend Formel F_{Sys} =

EDIV = 0: Teiler durch 8 für ECLK (Takt für 6800 Peripherie)
EDIV = 1: Teiler durch 16 für ECLK (Takt für 6800 Peripherie)

SLIMP = ***Anzeige*** für Referenzfrequenz (nicht programmierbar)
SLIMP = 0: Frequenzgenerator arbeitet mit externen Referenz (z.B. Quarz)
SLIMP = 1: interne Taktreferenz (Limp Mode) durch RC-Schaltung

SLOCK = ***Anzeige*** für Zustand der PLL Schaltung (nicht programmierbar)
SLOCK = 0: PLL Schaltung nicht eingerastet
SLOCK = 1: PLL Schaltung läuft stabil auf fester Frequenz

RSTEN = 0: bei Verlust der externen Referenz Limp Mode (Takt aus RC)
RSTEN = 1: bei Verlust der externen Referenz System Reset auslösen

STSIM = 0: Stromsparbetrieb (Befehl LPSTOP) ohne VCO Taktgenerator
STSIM = 1: Stromsparbetrieb (Befehl LPSTOP) mit VCO Taktgenerator

STEXT = 0: Stromsparbetrieb (Befehl LPSTOP) schaltet CLKOUT ab
STEXT = 1: Stromsparbetrieb (Befehl LPSTOP) mit Taktausgang CLKOUT

Das folgende Beispiel programmiert aus einer Quarzfrequenz von 32,768 kHz einen Systemtakt von 16,777 MHz.

```
    MOVE.W  #$7F03,SYNCR ; W=0, X=1, Y=63, EDIV=0
                         ; RSTEN=0, STSIM=1, STEXT=1
```

Das System Protection Control Register **SYPCR** auf der Byteadresse $FF FA21 legt die Arbeitsweise des Systemschutzes fest. Es kann ***nur einmal*** nach einem Einschalt- bzw. Tastenreset beschrieben werden. Zugriff nur im Systemstatus.

High-Byte (Byteadresse $FF FA20): wird nicht verwendet
Low-Byte (Byteadresse $FF FA21)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SWE	SWP	SWT	SWT	HME	BME	BMT	BMT
Reset:1	MODCLK	0	0	0	0	0	0

Die Bitpositionen haben folgende Bedeutung:
SWE = 0: Watchdog Timer gesperrt
SWE = 1: Watchdog Timer freigegeben (***Vorgabewert!***)

SWP = Vorteiler für Watchdogtimer (für MODCLK = High ist SWP = 0)
SWT = Teilerfaktor für Watchdogtimer

HME = 0: interner Haltmonitor gesperrt
HME = 1: interner Haltmonitor freigegeben

BME = 0: interner Busmonitor gesperrt
BME = 1: interner Busmonitor freigegeben

BMT = 0 0: interner Busmonitor wartet 64 Systemtakte
BMT = 0 1: interner Busmonitor wartet 32 Systemtakte
BMT = 1 0: interner Busmonitor wartet 16 Systemtakte
BMT = 1 1: interner Busmonitor wartet 8 Sytemtakte

Der Watchdog Timer (Wachhund) ist ein Zähler, der innerhalb einer programmierbaren Zeit durch einen Befehl (Software) neu gestartet werden muß. Geschieht dies nicht, so löst der Überlauf einen Reset, den Watchdogreset aus. Er ist nach einem Reset durch SWE = 1 zunächst freigegeben und wird in dem folgenden Beispiel gesperrt . Man beachte, daß das Systemschutzsteuerregister SYPCR nach einem Einschalt- bzw. Tastenreset nur ***einmal*** programmiert werden kann; der Wachhund und die Monitorfunktionen lassen sich ***nicht*** nach Belieben ein- bzw. ausschalten!

```
MOVE.B #$00,SYPCR  ; SWE = 0: Wachhund aus !!!
```

Neben dem Einschaltreset und dem Tastenreset gibt es weitere Resetzustände (*Bild 2-5*), die im Reset Status Register **RSR** auf der Byteadresse $FF FA07 angezeigt werden. Das Register kann nur im Systemstatus gelesen werden, die Bitpositionen werden von der Steuerung gesetzt.

High-Byte (Byteadresse $FF FA06): wird nicht verwendet
Low-Byte (Byteadresse $FF FA07)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EXT	POW	SW	HLT	0	LOC	SYS	TST

Die Bitpositionen werden auf 1 gesetzt, wenn der entsprechende Reset ausgelöst wurde. Sie haben folgende Bedeutung:

`EXT = 1`: Tastenreset durch Signal am Reseteingang

`POW = 1`: Einschaltreset beim Einschalten der Versorgungsspannung

`SW = 1`: Reset durch Watchdogtimer (Wachhund)

`HLT = 1`: Reset durch Haltmonitor

`LOC = 1`: Reset durch Ausfall des Referenztaktes

`SYS = 1`: Resetsignal durch RESET Befehl ausgegeben; kein Neustart!!!

`TST = 1`: Reset durch Testlogik (nur für Hersteller)

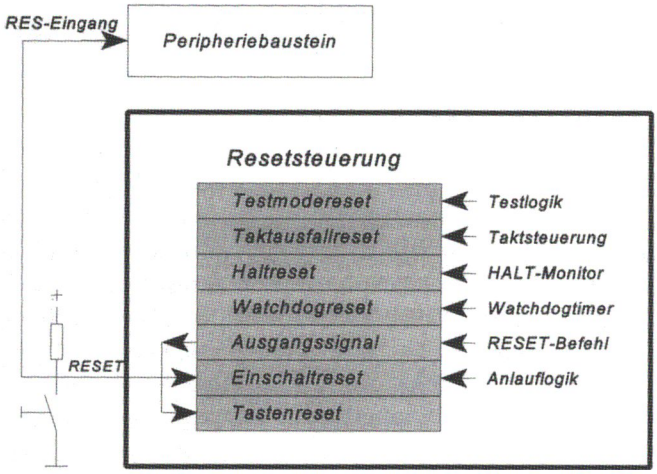

Bild 2-5: Die Resetsteuerung

Beim Anlauf des Systems nach einem Reset werden die Anfangswerte für den Systemstapelzeiger und den Befehlszähler aus dem externen Festwertspeicher gelesen, das Startprogramm programmiert anschließend die Systemsteuerregister. Die Adressierung des externen Speicherbausteins erfolgt immer mit dem Auswahlsignal CSBOOT, dessen Programmierung im Abschnitt 2.4 zusammen mit den anderen Freigabesignalen behandelt wird.

2.3 Die externe Bussteuerung

Dieser Abschnitt gibt einen Einblick in den Anschluß externer Speicher- und Peripheriebausteine. Für den Entwurf von Schaltungen sollten die Unterlagen der Hersteller herangezogen werden. *Bild 2-6* zeigt die Anschlußbelegung üblicher byteorganisierter Speicherbausteine.

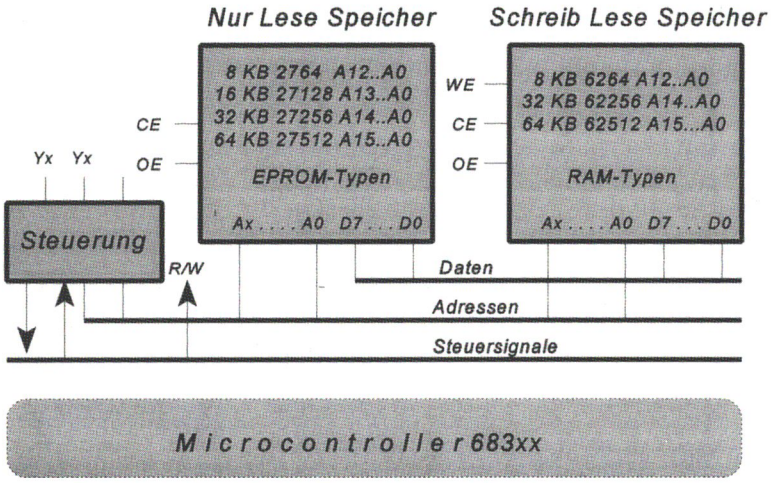

Bild 2-6: Byteorganisierte Speicherbausteine

Nur-Lese-Speicher (EPROM und EEPROM im Lesebetrieb) enthalten das auszuführende Programm. Der Steuereingang CE (Chip Enable = Freigabe des Bausteins) aktiviert im Zustand Low den internen Decoder zur Auswahl des Bytes, dessen Adresse an den Adreßeingängen A0 bis Ax anliegt. Der Steuereingang OE (Output Enable = Freigabe der Ausgangstreiber) schaltet im Zustand Low das ausgewählte Byte über die Datenausgänge D0 bis D7 auf den Datenbus.

Schreib-Lese-Speicher (RAM) enthalten die variablen (veränderlichen) Daten. Der Steuereingang CE aktiviert im Zustand Low den internen Decoder zur Auswahl des Bytes, dessen Adresse an den Adreßeingängen A0 bis Ax anliegt. Im Zustand Low des Steuereingangs WE (Write Enable = Schreibfreigabe) werden Daten in den Baustein geschrieben, im Zustand High aus ihm gelesen. Der Steuereingang OE schaltet beim Lesen im Zustand Low das ausgewählte Byte über die Datenausgänge D0 bis D7 auf den Datenbus; beim Schreiben hat er keine Funktion.

Die unteren Adreßleitungen von A0 bis Ax wählen je nach Speicherkapazität die Bytes auf den Bausteinen aus; die oberen Adreßleitungen geben über einen Decoder der ***Steuerung*** einen der angeschlossenen Bausteine frei. *Bild 2-7* zeigt den zeitlichen Verlauf der Signale beim Lesen und Schreiben.

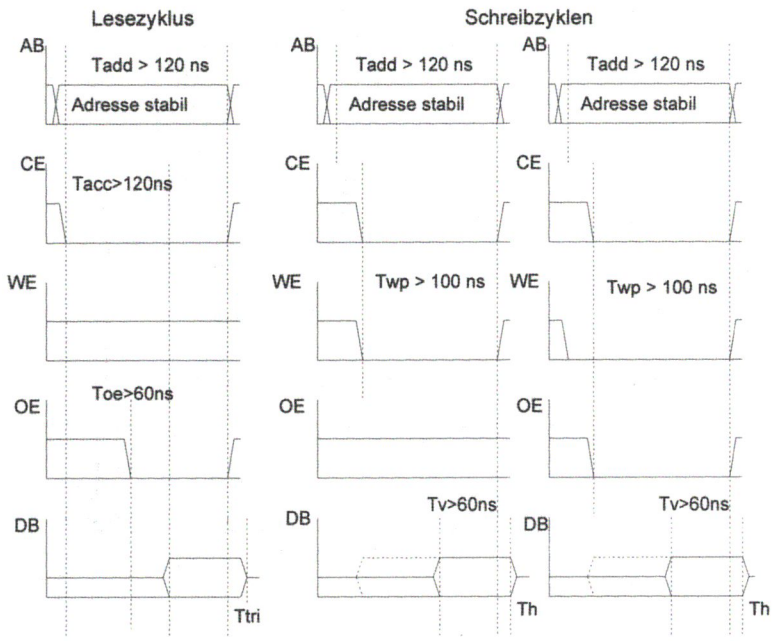

Bild 2-7: Lese- und Schreibzyklen (Zugriffszeit 120 ns)

In einem ***Lesezyklus*** muß die Adresse während der gesamten Zeit T_{add} stabil anliegen. Zum Auslesen werden CE ***und*** OE beide auf Low gelegt. Als Zugriffszeit T_{acc} bezeichnet man die Zeit von der Bausteinfreigabe mit CE bis zum Erscheinen der Daten auf den Datenbus; die Freigabezeit T_{oe} ist die Zeit von der Freigabe mit OE. Geht eines der beiden Steuersignale wieder auf High, so werden die Ausgänge wieder tristate. Legt man CE und OE gleichzeitig durch ein Decoderauswahlsignal auf Low, so gilt die längere Zugriffszeit T_{acc}. Legt man CE fest auf Low und schaltet nur OE mit der Bausteinauswahllogik, so gilt die kürzere Freigabezeit T_{oe}; jedoch steigt die Stromaufnahme.

In einem ***Schreibzyklus*** muß die Adresse während der gesamten Zeit T_{add} stabil anliegen. Bei Schreibzyklen, in dem OE dauernd inaktiv High ist, bestimmt die kürzere der beiden Low Zeiten von CE bzw. WE die Länge des Schreibimpulses T_{wp}. Bei Schreibzyklen, in denen OE zusammen mit CE auf Low geht, muß WE vorher auf Low liegen, um eine Datenausgabe zu verhindern. Werden die Daten mit der ersten steigenden Flanke von CE bzw. OE bzw. WE übernommen, so müssen sie während der Vorbereitungszeit T_v stabil anliegen; die Haltezeit T_h nach der Übernahmeflanke ist meist 0. Bei RAM Bausteinen verbindet man den Eingang WE mit dem Ausgang R/W (Read/Write = Lesen/Schreiben) des Controllers und gibt den Baustein mit CE *und* OE bzw. nur mit CE (OE = Low) frei.

Bild 2-8 zeigt die Signale des Controllers, mit denen die Speicher- und Peripheriebausteine bei externer Bussteuerung betrieben werden.

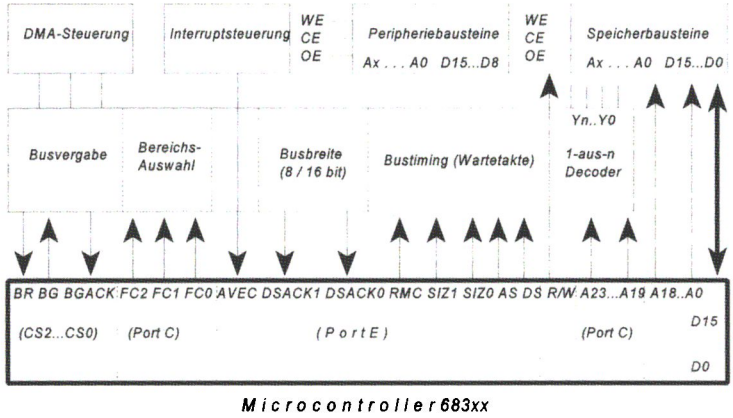

Bild 2-8: Die externe Bussteuerung

Die unteren *Adreßleitungen* A0 bis A18 sind immer verfügbar, die oberen Adreßleitungen A19 bis A23 können wahlweise auch als Datenport oder als Bausteinauswahlsignale programmiert werden. Der externe 1-aus-n Decoder bildet aus den höheren Adreßsignalen die Auswahlsignale CE der Bausteine.

Das *Richtungssignal* R/W (Read/Write = Lesen/Schreiben) ist in Lesezyklen High und in Schreibzyklen Low. Die Leitung ist immer verfügbar und wird meist direkt mit dem Eingang WE der RAM Bausteine verbunden.

Die *Timingausgänge* steuern den zeitlichen Ablauf der Datenübertragung zwischen dem Controller und den Bausteinen und leiten einen Buszyklus ein. Bei interner Bussteuerung können sie wahlweise auch als Datenport programmiert werden. Das Signal AS (Address Strobe = Adreßtastsignal) zeigt mit einem Zustand Low, daß sich gültige Adressen auf dem Adreßbus befinden. Das Signal DS (Data Strobe = Datentastsignal) zeigt in Schreibzyklen mit dem Zustand Low, daß der Controller gültige Daten auf den Datenbus gelegt hat. Mit dem Ausgang RMC (Read Modify Write Cycle = Lesen Verändern Zurückschreiben) zeigt der Controller unteilbare Zyklen an, die nicht unterbrochen werden können. Die Signale SIZE (Größe) kennzeichnen das Format der zu übertragenden Daten.

```
SIZ1 SIZ0
  0    0  : Byteübertragung (8 bit)
  0    1  : Wortübertragung (16 bit)
  1    0  : drei Bytes übertragen (Rest eines Langwortes)
  1    1  : Langwortübertragung (2 mal 16 bit)
```

Mit den *Timingeingängen* DSACK (Data Size Acknowledge = Bestätigung des Datenformats) bestimmt die externe Bussteuerung, wann und wie der Controller den Buszyklus beendet.

DSACK1	DSACK0	
1	1	: Controller muß Wartetakte einfügen
1	0	: beendet Buszyklus mit Byteübertragung
0	1	: beendet Buszyklus mit Wortübertragung
0	0	: reserviert (wirkt wie Wortübertragung)

Der *Interruptsteuereingang* AVEC (Autovector = Eigenvektor) meldet in einem Interruptzyklus, daß kein Vektor über den Datenbus geliefert wird, sondern daß die Interruptsteuerung einen eigenen Startvektor bereitstellen muß. Weitere Einzelheiten siehe Abschnitt 2.9 (Ausnahmeverarbeitung).

Mit den *Funktionsausgängen* FC (Function Code) läßt sich der Speicher in vier Blöcke zu je 16 MByte (24 Adreßleitungen) unterteilen. Der Ausgang FC2 entspricht dem S-Bit (System oder Benutzer) des Statusregisters SR. Der Ausgang FC1 unterscheidet zwischen einem Daten - und einem Programmzugriff.

FC2	FC1	FC0	
0	0	0	: reserviert für Hersteller
0	0	1	: Benutzer Datenbereich
0	1	0	: Benutzer Programmbereich
0	1	1	: reserviert für Hersteller
1	0	0	: reserviert für Hersteller
1	0	1	: System Datenbereich
1	1	0	: System Programmbereich
1	1	1	: CPU Bereich (Interruptzyklen)

Die *Busvergabesignale* BR (Bus Request = Bus anfordern), BG (Bus Grant = Bus bewilligen) und BGACK (Bus Grant Acknowledge = Busbewilligung bestätigen) werden nur in Sonderfällen verwendet, wenn z.B. eine DMA Steuerung (Direct Memory Access = direkter Speicherzugriff) den Controller in den Tristatezustand versetzt, um direkt auf den Speicher zugreifen zu können.

Die beiden folgenden Bilder zeigen stark vereinfachte Abläufe des Lese- und Schreibvorganges. Die angegebenen Zeiten beziehen sich auf einen Systemtakt von 16,777 MHz entsprechend einer Taktzeit von 59,6 ns. Der Controller *leitet* in den Takten 1 und 2 (Halbtakten S0 bis S3) den Buszyklus nur *ein*; die externe Bussteuerung *beendet* ihn mit den DSACK Signalen, die in der Mitte des zweiten Taktes abgetastet werden. Sind beide Steuereingänge am Ende von S2 High, so schiebt die interne Steuerung zusätzliche Wartetakte ein, um die Zugriffszeit der Bausteine zu verlängern. Dieser *asynchrone* Busbetrieb wird mit dem Systemtakt synchronisiert. Der letzte Takt (Halbtakte S4 und S5) macht die Steuersignale inaktiv und beendet den Buszyklus. Die SIZE Ausgänge geben aufgrund des auszuführenden Befehls die Bitbreite der Daten an; mit den DSACK Ein-

gängen meldet die externe Bussteuerung, ob die Übertragung byteweise (8 bit) oder wortweise (16 bit) erfolgt. Durch diese *dynamische Busstruktur* kann ein Wort (16 bit) in einem Zyklus oder in zwei Zyklen übertragen werden; ein Langwort (32 bit) in zwei oder vier Zyklen.

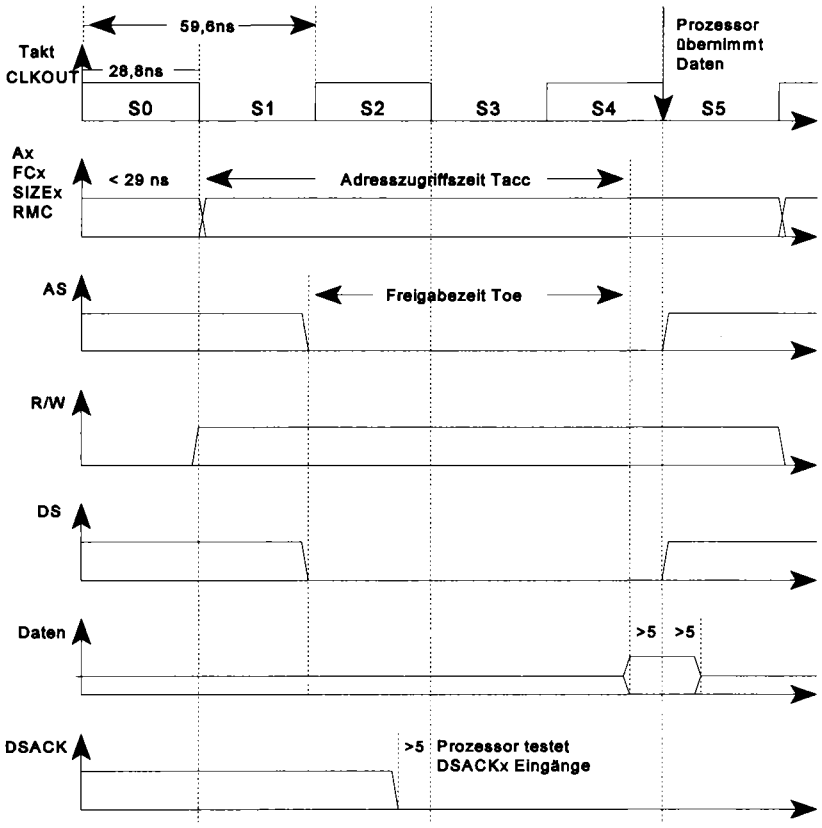

Bild 2-9: Asynchroner Lesezyklus bei 16,777 MHz ohne Wartetakte

In einem ***Lesezyklus*** (*Bild 2-9*) sind die Adressen am Ende des 1. Halbtaktes S0 stabil; R/W ist High. Am Ende des 2. Halbtaktes S1 werden die Zeitsteuersignale AS und DS aktiv Low, die zusammen mit einem Adreßdecoder den zu lesenden Baustein auswählen. Ohne Wartetakte (DSACK am Ende des Halbtaktes S2 Low) wird der Lesezyklus durch Lesen des Datenbus am Ende des Halbtaktes S4 beendet. Die Daten müssen mindestens 5 ns vorher (Vorbereitungszeit) und 5 ns nachher (Haltezeit) stabil sein. Am Ende des letzten Halbtaktes S5 sind die Steuersignale inaktiv, Adreß- und Datenbus sind undefiniert. Die minimalen Zugriffszeiten berechnen sich aus der Anzahl der Wartetakte WS und der Zykluszeit T_{zyk} des Systemtaktes zu:

$T_{acc} = (2,5 + WS) * T_{zyk} - 34$ (Adressen stabil während des Lesens)
$T_{oe} = (2,0 + WS) * T_{zyk} - 34$ (Fallende Flanke AS und DS bis Daten gültig)

WS	T_{acc}	T_{OE}
0	115ns	85ns
1	175ns	145ns
2	232ns	204ns

Für WS = 0 und T_{zyk} = 59,6 ns ergeben sich für die zu lesenden Speicherbausteine Zugriffszeiten von T_{acc} = 115 ns und T_{oe} = 85 ns.

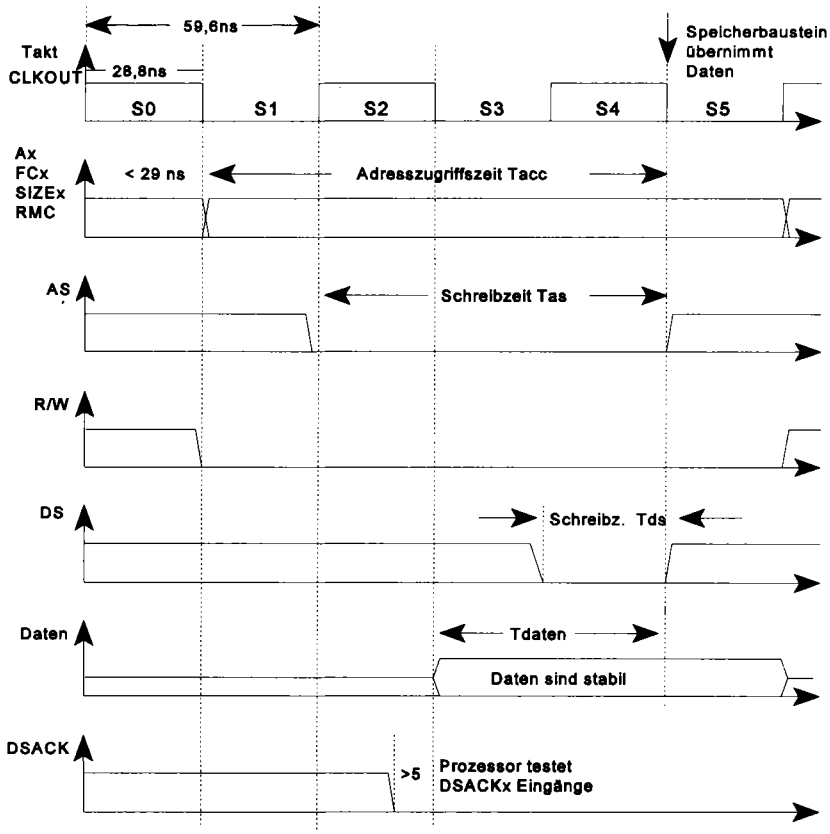

Bild 2-10: Asynchroner Schreibzyklus bei 16,777 MHz ohne Wartetakte

In einem ***Schreibzyklus*** (*Bild 2-10*) sind die Adressen am Ende des 1. Halbtaktes S0 stabil; R/W ist Low. Am Ende von S1 wird das Zeitsteuersignal AS aktiv Low, das Zeitsteuersignal DS erst einen Takt später am Ende von S3, wenn der Controller gültige Daten auf den Datenbus gelegt hat. Ohne Wartetakte (DSACK am Ende des Halbtaktes S2 Low) wird der Schreibzyklus mit dem Halbtakt S4 beendet. Am Ende des letzten Halbtaktes S5 sind die Steuersignale inaktiv,