V

HARDWARE HANDBUCH

MICROCOMPUTER FAMILIE

MCS 6500

Copyright by COMMODORE - MOS TECHNOLOGY

COMMODORE - MOS TECHNOLOGY verfügt über alle Rechte der deutschen und englischen Ausgabe.

Nachdrucke und Vervielfältigungen, auch auszugsweise, sind nur mit ausdrücklicher Genehmigung von COMMODORE - MOS TECHNOLOGY gestattet.

COMMODORE BÜROMASCHINEN GMBH
Frankfurter Straße 171–175 · D - 6078 Neu-Isenburg · Postfach 4 26
Telefon (0 61 02) 80 03 · Telex 04 185663 como d

VORWORT

Das MOS Technology Inc. MCS 6500 Mikrocomputer System vereinigt die besten Eigenschaften einer Familie in der zweiten Generation zu einer Produktreihe, die in Preiswürdigkeit und Leistung eine führende Stelle einnimmt. Eine wachsende Gruppe von Produkten und eine einzigartige Mikroprozessorfamilie bieten dem Kunden Lösungen für die komplexen Entwurfsprobleme, die sich heute Programmierern und Entwicklern stellen.

Verfahren zur Herstellung integrierter Schaltungen haben die Mikroprozessoren in die vorderste Linie komplizierter und hochentwickelter Bauteile gerückt. Die MCS 6500 Familie nutzt eine hochentwickelte, aber erprobte Prozesstechnologie. Für die Erreichung der guten Leistungsdaten der MCS 6500 Familie im 5 Volt-Betrieb sind N-Kanal Silizium Gate und Depletion Loads die Grundlage.

Durch die Verwendung von Depletion Loads erreicht die N-Kanal Silizium Gate Technologie größere Arbeitsgeschwindigkeiten, kleinere Leistungsaufnahme und kleinere Chipgröße als vorhergehende Herstellungsverfahren. Diese Eigenschaften bieten ein Preis-Leistungsverhältnis, das die MCS 6500 Familie zu einem Produkt macht, das den wirtschaftlichen und technischen Anforderungen heutiger Systeme in hohem Maße entspricht.

Da das Konzept der "Mikroprozessor Familie" in der Industrie einzigartig ist, ist eine kurze Erläuterung der MCS 6500 Produktreihe angebracht. Es ist nützlich, die grundlegende Produktstruktur der MCS 6500 Familie zu verstehen.

Die Mikroprozessor Familie besteht aus der Serie MCS 65XX. Innerhalb dieser Familie bietet eine Reihe von 8-Bit Einheiten dem Kunden einen weiten Bereich von Wahlmöglichkeiten und Fähigkeiten.

GESAMTHOCHSCHULBIBLIOTHEK * PADERBORN *

79/10213

Für die Anwendung steht eine breite Auswahl an Einheiten zur Verfügung, aus der der Kunde die beste Lösung für seine Anwendung auswählt. Das Konzept der „Mikrocomputer Familie" hat jedoch erheblich größere Wirkung für den Kunden mit zahlreichen Anwendungen. Dabei wird jeder dieser Anwendungen durch ein besonderes Mitglied der Familie Rechnung getragen. Für diese Anwender ist es wichtig, daß die verschiedenen Mikroprozessoren seiner Wahl kompatibel sind - in der Hardware (hinsichtlich der Busse und der elektrischen Daten) und in der Software. Die MCS 65XX Produkte stellen die erste Mikroprozessor-Familie dar, die ein solches Maß an Kompatibilität bietet. Sie wurde als Familie geplant, die vollkommen Software-und Hardwarekompatibel ist und damit dem Entwickler einen Leistungsbereich bietet, aus dem er auswählen kann. Der MCS 6502 mit 40 Stiften bietet 65 K - Byte adressierbaren Speicher. Der MCS 6503, MCS 6504 und MCS 6505 sind die ersten Versionen mit 28 Stiften und verschiedenen Wahlmöglichkeiten der Adressierfähigkeit und Steuerfunktionen.

Die Reihe MCS 652X besteht aus peripheren Ein/Ausgangseinheiten (I/O). Der MCS 6520 ist ein direkter Ersatz für den Motorola MC 6820, ein peripherer Interface Adapter (PIA). Weitere Mitglieder dieser Reihe beinhalten Erweiterungen der I/O Möglichkeiten.

Die MCS 653X umfaßt die Kombinationseinheiten. Sie bestehen aus verschiedenen Kombinationen von RAM, ROM, I/O und Timing. Der MCS 6530 enthält zum Beispiel 1K Byte ROM, 64 Byte RAM, einen Intervall Timer und 16I/O Leitungen. Die folgenden Produkte dieser Serie bieten dem Kunden verschiedene Möglichkeiten und Realisierungen von I/O Timing und Speicher.

Die Reihe MCS 654X umfaßt Lesespeicher (ROM). Sie sind zugeschnitten auf die Anforderung beim Speichern umfangreicher Programme, wie sei bei vielen Anwendungen der 6500 Familie vorkommen. Der erste Speicher ist ein 16 K (2K x 8) ROM, der MCS 6540.

Alle Teile der Serie MCS 6500 stimmen in Herstellungstechnik und elektrischen Daten überein. Der Entwickler erhält mit dieser Familie kompatibler Produkte alle notwendigen Teile, um ein den höchsten Anforderungen genügendes System zusammenzustellen.

Die MCS 6500 Familie wird ergänzt durch eine Auswahl Schreib-Lesespeicher (Random Access Memories, RAM), die vollständig mit der Mikrocomputer Familie kompatibel sind. Die ersten Bausteine sind der MCS 6102, mit dem 2102 austauschbar, und der MCS 6111, mit dem 2111 austauschbar.

Alle MCS 6500 Teile sind mit der M 6800 Busstruktur kompatibel, um kleinste I/O Kosten und größte Flexibilität zu gewährleisten. Kapitel 1 dieses Handbuchs führt den Leser in das MCS 6500 Mikrocomputer System ein. Es umfaßt eine Einführung in die Terminologie, eine Erklärung der Systemteile eines Mikrocomputer Systems im allgemeinen, und bespricht dann die Teile der MCS 6500 Familie.

Kapitel 2 ist anwendungsbezogen. Es bespricht die Systemkonfiguration, den I/O Port, das Handshaking und gibt genaue Beispiele für Interrupt Prioritäten, Interfacing Peripherien, Verfahren der direkten Speicheradressierung und der Steuerung im System.

Kapitel 3 zielt auf die wichtige Aufgabe des Systemaufbaus. Es führt den Leser Schritt für Schritt durch eine statische und dynamische Analyse der Grundelemente des Systems um den reibungslosen Übergang vom geplanten zum arbeitenden System zu ermöglichen.

INHALTSVERZEICHNIS

Kapitel 1 - Das Mikrocomputer-System MCS 6500		1
1.0	Entwickeln mit Mikrocomputer-Systemen	3
1.1	Einführung in Mikrocomputer-Systeme	4
1.1.1	Organisation eine Mikrocomputer-Systems	4
1.1.2	Grundsätzliche Arbeitsweise	4
1.1.3	Begriffe und Konzepte zur Adressierung	6
1.1.3.1	Bit	6
1.1.3.2	Adressbereich	6
1.1.3.3	Adress-Seite (PAGE)	7
1.1.4	Bauteile des Systems	7
1.1.4.1	Taktgenerator	9
1.1.4.2	Programmspeicher	9
1.1.4.3	Datenspeicher	10
1.1.4.4	Eingangs/Ausgangs-Bausteine	11
1.1.4.5	Der Mikroprozessor	12
1.2	Einführung in die Mikroprozessor-Familie MCS 650X	14
1.2.1	Der MCS 6502	14
1.2.2	Der MCS 6503, MCS 6504 und MCS 6505	15
1.3	MCS 6500 System Konzepte	16
1.3.1	Bus Struktur	16
1.3.2	Prozessor Interrupts	17
1.3.2.1	Anwendungen für Interrupts	19
1.3.2.2	Setzen der Interrupt Prioritäten	25
1.3.2.3	System-Verbindung für Interrupts	26
1.3.2.4	Bedienung des Interrupts	26
1.3.2.5	Interrupt-Anforderung ($\overline{IRQ}$)	27
1.3.2.6	Nichtmaskierbarer Interrupt ($\overline{NMI}$)	30
1.3.3	Rücksetzen des Systems (System Reset)	31

1.4	Die Mikroprozessoren	33
1.4.1	Der MCS 6502	33
1.4.1.1	Einführung	33
1.4.1.2	Anschlußbelegung des MCS 6502	37
1.4.1.2.1	Vcc, Vss - Stromversorgungs-Leitungen	37
1.4.1.2.2	AB00 - AB15 - Adressbus	37
1.4.1.2.3	DB0 - DB) - Daten-Bus	39
1.4.1.2.4	R/W - Schreiben/Lesen (Read/Write)	40
1.4.1.2.5	Das SYNC-Signal	41
1.4.1.2.6	RDY - Fertig (Ready)	41
1.4.1.2.7	$\overline{\text{NMI}}$ - Nicht-maskierbarer Interrupt (Non-maskable Interrupt)	43
1.4.1.2.8	$\overline{\text{IRQ}}$ - Interrupt-Anforderung (Interrupt Request)	43
1.4.1.2.9	$\overline{\text{RES}}$ - Rücksetzen (RESET)	46
1.4.1.2.10	Takt-Erzeugung	47
1.4.1.2.11	S.O. - Setzen des Überlaufs (Set Overflow)	48
1.5	Baustein für periphere Schnittstelle MCS 6520	56
1.5.1	Einführung	56
1.5.2	Organisation des MCS 6520	57
1.5.2.1	Dateneingangs Register	60
1.5.2.2	Steuerregister (CRA und CRB)	60
1.5.2.3	Datenrichtungs-Register (DDRA, DDRB)	61
1.5.2.4	Periphere Ausgangs-Register (ORA, ORB)	61
1.5.2.5	Steuerung des Interrupt-Status	62
1.5.2.6	Puffer für periphere Schnittstelle (A,B) und Datenbus-Puffer (DBB)	62
1.5.3	Schnittstelle zwischen MCS·6520 und MCS 650X	62
1.5.3.1	Datenbus (D0 - D7)	62
1.5.3.2	Einschalt-Eingang (E, Enable)	63
1.5.3.3	Schreiben/Lesen (R/W, Read/Write)	63

1.5.3.4	Leitungen zu Bausteinauswahl (CS1, CS2, $\overline{CS3}$)	63
1.5.3.5	Register-Auswahlleitungen (RS 0, RS 1)	65
1.5.3.5.1	Auslesen des peripheren Eingangs/ Ausgangs-Kanals A	66
1.5.3.5.2	Auslesen des peripheren Eingangs/ Ausgangs-Kanals B	67
1.5.3.6	Rücksetzen ($\overline{RES}$, Reset)	67
1.5.3.7	Anforderungs-Leitung für Interrupt ($\overline{IRQA}$, $\overline{IRQB}$)	71
1.5.3.7.1	Steuerung von $\overline{IRQA}$	71
1.5.3.7.2	Steuerung von $\overline{IRQB}$	72
1.5.4	Schnittstelle zwischen MCS 6520 und Peripherie-Geräten	73
1.5.4.1	Periphere Eingangs/Ausgangs-Kanäle	73
1.5.4.1.1	Peripherer Eingangs/Ausgangs-Kanal A (PA 0 - PA 7)	73
1.5.4.1.2	Peripherer Eingangs/Ausgangs-Kanal B (PB 0 - PB 7)	74
1.5.4.2	Interrupt-Eingang/Peripherie-Steuerung (CA 1, CA 2, CB 1, CB 2)	74
1.5.4.2.1	Peripherie A: Interrupt-Eingang/ Peripherie-Steuerung (CA 1, CA 2)	75
1.5.4.2.2	Peripherie B: Interrupt-Eingang/ Peripherie-Steuerung (CB 1 , CB 2)	76
1.5.5.2	Arbeitsweise des MCS 6520 in MCS 6500 Systemen	79
1.6	Baustein für periphere Schnittstelle/ Speicher - MCS 6530	80
1.6.1	Einführung	80
1.6.2	Anschlußbelegung	80
1.6.2.1	Rücksetzen ($\overline{RES}$)	82
1.6.2.2	Eingangstakt	82
1.6.2.3	Schreiben/Lesen (R/W)	82
1.6.2.4	Anforderung eines Interrupt ($\overline{IRQ}$)	82
1.6.2.5	Datenbsu (D 0 - D 7)	84

1.6.2.6	Periphere-Datenkanäle	84
1.6.2.7	Adressleitungen (A 0 - A 9)	84
1.6.3	Interne Organisation	86
1.6.3.1	ROM - 1 K Byte (8 K Bits)	86
1.6.3.2	RAM - 64 Bytes (512 Bits)	87
1.6.3.3	Interne Peripherie - Register	87
1.6.3.4	Zeitgeber	87
1.6.4	Adressierung	89
1.6.4.1	Adressierung für einen Baustein	94
1.6.4.2	Adressierung für sieben Bausteine	94
1.6.4.3	Adressierung des Eingangs/Ausgangs-Registers und des Zeitgebers	95

Kapitel 2 - Die Konfiguration des Mikrocomputer-Systems

2.1	Die Aufgabe der System-Konfiguration	96
2.2	Eingangs/Ausgangs-Verfahren	98
2.2.1	Der Eingangs/Ausgangs-Kanal für allgemeine Anwendung	98
2.2.2	Der Peripherie-Schnittstellen-Baustein für besondere Anwendungen	99
2.2.3	Konfiguration des Mehrzweck-E/A-Kanals	100
2.2.3.1	Zuordnung von Ausgängen	101
2.2.3.2	Zuordnung von Eingängen	103
2.2.4	Einschalten der Stromversorgung	104
2.2.5	Handshaking	108
2.2.5.1	Handshaking für Datenübertragungen vom Prozessor	108
2.2.5.2	Handshaking bei Datenübertragungen in den Prozessor	111
2.3	Konfiguration der Schnittstelle zwischen Mikroprozessor und Unterstützungs-Bausteinen	113
2.3.1	Zuordnung der Adressen im System MCS 6500	113
2.3.1.1	Zuweisung der Adressen für ROM	115

2.3.1.2	Zuordnung der Adressen für RAM	116
2.3.2	Weitere Verfahren zur Zuordnung von Adressen	119
2.3.3	Interrupts	119
2.3.3.1	Feststellen der Interrupt-Priorität	121
2.3.3.2	Beispiel 1: Auswahl des Interrupt-Vektors	121
2.3.3.3	Beispiel 2: Benutzung der Software-Möglichkeiten des Prozessors	122
2.3.4	Die Anwendung von RDY zur Steuerung der Speicher-Schnittstelle	126
2.3.4.1	Schnittstelle mit langsamen PROM	126
2.3.4.2	Verfahren zum direkten Speicher-Zugriff (DMA, Direct Memory Access)	128
2.3.4.3	Steuerung von dynamischen RAM im System MCS 6500	129
2.4	Weitere System-Betrachtungen	132
2.4.1	Peripherie-Schnittstellen-Bausteine	132
2.4.2	RAM	133
2.4.3	ROM	134
2.5	Untersuchung der Leistung des Systems	135

Kapitel 3 - Inbetriebnahme des Mikrocomputer-Systems MCS 6500

3.0	Einführung in die Prüfung von Mikro-computern	137
3.1	Statische Prüfung	138
3.1.1	Einführung	138
3.1.2	Ausführung eines Einzelzyklus	138
3.1.3	Ausführung eines einzelnen Befehls	142
3.2	Dynamische Prüfung	142
3.2.1	Einführung	142
3.2.2	Extern hervorgerufene Schleifen	145
3.2.3	Software - Schleifen	147
3.3	Systemdiagnose mit Hardware Hilfs-mitteln für den Programmierer	149

3.3.1		151
3.3.2	TIM - Monitor für Fernschreibeingang	153
3.4	Verfahren zur Inbetriebnahme des Mikroprozessors	156
3.4.1	Einführung	156
3.4.2	Stromversorgung des Systems - Schritt 1	157
3.4.3	Grundlegende Zeitbeziehung im System - Schritt 2	157
3.4.4	Rücksetzen des Systems - Schritt 3	158
3.4.4.1	Statische Untersuchung von Systemeinzelheiten	158
3.4.4.2	Dynamische Untersuchung von System-Einzelheiten	160
3.4.4.2.1	Oberprüfung des Adress-Bus	164
3.4.4.2.2	Oberprüfung des Datenbus	164
3.4.5	Gebaue Untersuchung von Einzelteilen	166

ANHANG

ABBILDUNGEN

Kapitel 1 - Das MCS Mikrocomputer System

1.1	Organisation des Mikrocomputer-Systems	5
1.2	Adressbus und Bezeichnung zum Speicherfeld	8
1.3	Teil einer ROM-Matrix	10
1.5	Zeit-Tabelle für Takt und Schreiben/Lesen	18
1.6	Zeitdiagramm des Zweiphasen-Taktes	20
1.7	Zeitdiagramm für Lesen aus Speicher oder Peripherie	21
1.8	Zeitdiagramm für Schreiben in Speicher oder Peripherie	22
1.9	Interrupts wired -or- Hardware-Konfiguration von Peripherie-Schnittstelle	27
1.10	Sequenz zur Bedienung von $\overline{IRQ}$	29
1.11	MCS 650X interner Aufbau	34
1.13	MCS 650X Zeitdiagramm des Systems	38
1.14	Beispiele der Interrupt-Erkennung beim MCS 650X	44
1.15	MCS 6502 Beschreibung der Stiftbelegung	49
1.16	MCS 6502 Taktgenerator, Quarz in Parallel-resonanz	50
1.16a	MCS 6502 Taktgenerator mit Quarz in Serien-resonanz	50
1.16b	MCS 6502 Taktgenerator mit Quarz in Parallel-resonanz	50
1.17	MCS 6502 Taktgenerator mit RC-Glied	50
1.18	MCS 6502 SYNC-Signal	51
1.19	MCS 6503, 6504, 6505 Funktionsbeschreibung	52
1.20	MCS 6503, 6504, 6505 Stiftbelegung	53
1.21	MCS 6503, 6504, 6505 Taktgenerator mit Quarz in Serienresonanz	54
1.22	MCS 6503, 6504, 6505 Taktgenerator mit Quarz in Parallelresonanz	54

1.23	MCS 6520 Schaltbild der Schnittstelle	55
1.24	MCS 5420 Stiftbelegung des peripheren Interface Adapters	58
1.25	MCS 6520 innerer Aufbau	59
1.26a	Mikroprozessor Zeitdiagramm Lesen	64
1.26b	Mikroprozessor Zeitdiagramm Schreiben	64
1.27a	Peripherie A - Zeitdiagramm	68
1.27b	Peripherie B - Zeitdiagramm	69
1.28a	Peripherie E/A-Kanal A Puffer	70
1.28b	Peripherie E/A-Kanal B Puffer	70
1.29	Steuerregister Bit-Beschreibung	77
1.30	Steuerung der Interrupt-Eingänge	77
1.31a	Steuerung von CA 2 (CB 2) als Interrupt-Eingang (Bit 5 = "0")	77
1.31b	Steuerung der CA 2 Ausgangs-Arten	78
1.31c	Steuerung der CB 2 Ausgangs-Arten	78
1.32	MCS 6530 Beschreibung der Stiftbelegung	81
1.33	MCS 6530 innerer Aufbau	85
1.34	Grundbestandteile des Teitgebers	88
1.35	Beispiel eines Interrupt, ausgelöst durch den Zeitgeber	90
1.36	MCS 6530 Kodier-Diagramm für 1 Baustein	91
1.37	MCS 6530 Adressen-Schema für 7 Bausteine	92
1.38	Adressier-Dekodierung für E/A-Register und Zeitgeber	93

Kapitel 2 - Konfiguration des Mikrocomputer-Systems

2.1	Steuerung des untergeordneten Bits im MCS 6520 Ausgabe-Register	102
2.2	MCS 6520 Steuerung der mit Transistor geschalteten Magnetspulen	105
2.3a	MCS 6520 Steuerung eines PNP-Treibertransistors für eine Magnetspule	105
2.3b	MCS 6520 Steuerung der Versorgungsspannung und der Magnetspulen	106
2.4	MCS 6520 Treiben der TTl-Puffer	106

2.5	MCS 6520 Steuerung der Magnetspulen mit Enable-Signalen und TTL-Interface	106
2.6	Handshake-Sequenz für Schreiben	109
2.7	Handshake-Sequenz für Lesen	110
2.8	Organisation des Mikrocomputer-Systems	114
2.9	Beispiel der "AND"-Funktion unter Anwendung der Adressleitungen	116
2.10	Typische Adresszuweisung	117
2.11	Null-Seite, Adressierschema zur Baustein-Auswahl	118
2.12	Auswahl des Interrupt-Vektors	120
2.13	Anwendung des MCS 6520 für indirekte Interrupt Routinen (Jump)	123
2.14a	Prioritätsencoder verbunden mit den unteren Bits des 6520	124
2.14b	Verbindung von Prioritätsencoder zur peripheren Schnittstelle	125
2.15	Software-Programm zum implementieren des Interrupts in obiger Hardware-Konfiguration	125
2.16	Schnittstelle für langsame PROMs	127
2.17	Logik zum Erzeugen des Signals "Bus verfügbar" für DMA Anwendungen	127
2.18	Steuer-Logik für Refresh-Signal für dynamische RAMs	131

Kapitel 3 - Aufbau des MCS 6500

3.1	Vorgeschlagene Steuer-Logik für statische Prüfung	139
3.2	Zeitdiagramm für Einzelzyklus	140
3.3	Datenfalle für Einzelzyklus	143
3.4	Ausführung eines Einzelbefehls	144
3.5	Vorgeschlagene Konfiguration für Test des dynamischen RESET	146
3.7	Adressleitungen im MCS 650X System	162
3.7a	Richtige Adressenleitungen	162
3.7b	Adressenleitungen mit Oberlast	162
3.8	Der Datenbus im MCS 650X System	163

KAPITEL 1

DAS MIKROCOMPUTER-SYSTEM MCS 6500

Die vergangenen Jahre waren die Entwicklung eines aufregenden neuen Konzeptes in der Entwicklung der Elektronik. Die bisherige Art, Systeme zu entwickeln, wird durch den einchip programmierbaren Mikroprozessor überholt. Zu Beginn war der Mikrocomputer ein verhältnismäßig einfacher, aber schwierig zu benutzender programmierbarer Baustein, der einfache Steuer- oder Rechenprobleme lösen konnte. Inzwischen ist er jedoch zu einem leistungsfähigen, preiswerten und einfach zu benutzenden Baustein geworden, der fast alle Systeme steuern kann.

Drei hauptsächliche Eigenschaften von Systemen mit Mikroprozessoren sind der Grund dieser Revolution.

1. Mikroprozessoren ermöglichen eine bedeutende Verkleinerung der Gesamtkosten des Systems für Produkte, die augenblicklich gefertigt werden. Viele Hersteller können durch neue Planung ihrer Produkte um einen Mikroprozessor einen Preisvorteil entwickeln oder gegenüber der Konkurrenz halten.

2. Die fallenden Kosten für Mikrocomputer-Systeme eröffnen große neue Märkte für Mikroprozessoren.Jetzt werden mit den modernen, preiswerten Mikroprozessoren eine große Zahl von Systemen auf den Markt gebracht, die vorher unmöglich oder bestenfalls unpraktikabel gewesen wären.

3. Während der Preis für Mikroprozessoren fällt, nehmen deren Möglichkeiten schnell zu. Sie können daher in mehr Systeme als je zuvor eingeplant werden.

Wer heute ein Produkt neu entwickelt oder dessen Kosten senken möchte sollte sich fragen, ob nicht ein Mikroprozessor sein Problem löst. Der Erfolg des Mikroprozessors beruht auf der Tatsache, daß Entwicklungsingenieur und Programmierer durch Verwendung von kostenwirksamen integrierten Schaltungen ihre Erfahrung bei der Lösung einer Vielzahl von Entwicklungsproblemen anwenden können. Einfache bis sehr schwierige Entwicklungsaufgaben können durch Zusammenstellung einer kleinen Zahl von großen integrierten Schaltungen gelöst werden.

Wenn dieselben integrierten Schaltungen in einer Vielzahl einzigartiger Entwürfe verwendet werden, stellt sich die Frage was sie einzigartig macht. Die Antwort darauf heißt: die Programmierung. Obwohl viele verschiedene Entwicklungen eine gemeinsame Hardware haben können, hat jede ihr eigenes Programm. Das führt uns zu einer weiteren wichtigen Eigenschaft von Mikrocomputern, den Festwertspeicher. Diese integrierte Schaltung, ein " Read-Only Memory " (ROM) speichert das Systemprogramm. Der Halbleiter Hersteller kann das Muster, das die Daten im ROM festlegt, auf einfache Weise erzeugen. Deswegen sind die Kosten für die "Entwicklung" eines ROM im allgemeinen kleiner als 10 % der Kosten für die Entwicklung eines speziellen Bausteins für den Kunden. Der Anwender hat den Vorteil von Standarderzeugnissen, die in hohen Stückzahlen gefertigt werden, obwohl sein System für seine eigene Anwendung einzigartig ist, da ein Baustein dieses System "maßgeschneidert" ist.

1.0 ENTWICKELN MIT MIKROCOMPUTER-SYSTEMEN

Viele Fachleute werden überrascht feststellen, daß die Entwicklung eines Systems rund um einen Mikroprozessor sehr ähnlich der Entwicklung eines Systems mit herkömmlicher Logik ist. Die Art des Vorgehens ist die Gleiche, nur die Ausführung der einzelnen Schritte weicht ab.

Ein kurzer Blick auf den Entwurfsvorgang eines Systems rückt die Mikrocomputer-Entwicklung in die richtige Perspektive. Damit wird auch der Zweck dieses Handbuches erläutert. Bei der Entwicklung eines Systems gibt es folgende Schritte:

1. Anforderungen an das System festlegen. Welche Funktionen soll es ausführen?
2. Grundbauteile des Systems festlegen.
3. Einzelheiten des Entwurfs vervollständigen.
4. Bau und Prüfung von Prototypen
5. Abschluß der Entwicklung und Beginn der Produktion.

Schritt 1 gilt für jedes System und im allgemeinen für jedes Produkt. Schritt 2 bildet des Ausgangspunkt für eine Entwicklung mit Mikroprozessoren. Bei sehr kostenempfindlichen Anwendungen ist ein sorgfältiger Blick auf die gesamten Systemkosten nötig. Kann ein Mikroprozessor die Aufgabe innerhalb der Preisgrenzen lösen. Im entgegengesetzten Fall müssen die Fähigkeiten des Mikroprozessors genau untersucht werden, um sicherzugehen, daß die verfügbaren Bausteine die nötige Funktion tatsächlich ausführen können. Ist ein Mikroprozessor schnell genug, um das System zu betreiben? Ist mehr als ein Prozessor nötig?

Mit diesem Handbuch soll der Entwickler lernen ein Mikroprozessor-System zusammenzustellen und die Leistung des Systems festzustellen. Nach diesem Schritt wird der Entwurf durch die Entwicklung des System-Programms vervollständigt. Die Ausführung des Systemprogramms wird im Programmierhandbuch besprochen.

1.1 Einführung in Mikrocomputer-Systeme

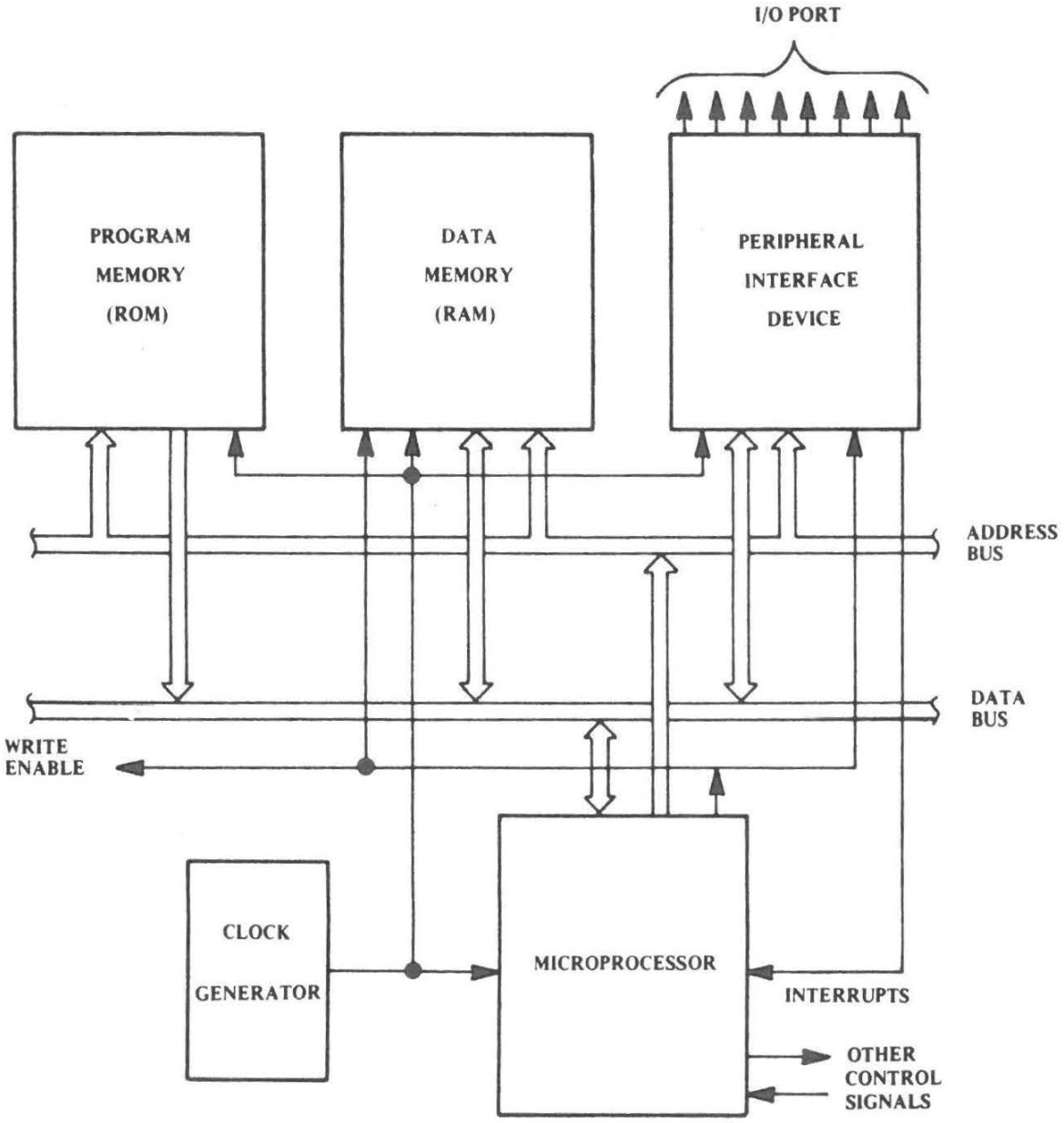
1.1.1 Organisation eines Mikrocomputer-Systems

Bild 1.1 zeigt den grundlegenden Aufbau eines Mikrocomputer-Systems. Der Entwickler muß die Arbeitsweise jedes einzelnen Bauteils und jedes Datenweges im System verstehen. Das wird im Nachfolgenden im einzelnen besprochen. Außerdem wird die Arbeitsweise des Gesamtsystems und die Benutzung der verschiedenen Signalwege beschrieben.

1.1.2 Grundsätzliche Arbeitsweise

Der Mikrocomputer kann als System beschrieben werden, daß in seinen Einzelheiten sehr einfach ist und in seiner Gesamtarbeitsweise sehr kompliziert. Es führt komplizierte Aufgaben dadurch durch, daß es eine große Zahl von einfachen Arbeitsgängen ausführt. Die Steuerung des System ist hauptsächlich Aufgabe des Prozessors. Durch Ausgeben der Adressen für den Programmspeicher wird die Reihenfolge der Arbeitsgänge gesteuert. Der Prozessor steuert die tatsächlichen Arbeitsgänge im System durch Auswertung und Ausführung der Befehle die er vom Programmspeicher erhält. Der Prozessor ist bei weitem der komplizierteste Baustein im System. Aus diesem Grund ist es für die gesamten Systemkosten wichtig, daß dieses Bauteil für viele verschiedene Anwendungen das gleiche bleibt. Auf diese Weise können die recht hohen Entwicklungskosten zwischen Tausenden von Anwendern geteilt werden. Zusätzlich haben diese Anwender den wirtschaftlichen Vorteil einer Produktion in großen Stückzahlen.

Der Prozessor veranlaßt das System durch Lesen des ersten Befehls im Programm zu den gewünschten Arbeitsgängen und führt die sehr einfache Aufgabe aus, die durch das besondere Muster der Bits in diesem Befehl festgelegt ist (Ausführung dieses Befehls).



Organization of Microcomputer System

ABBILDUNG 1.1: ORGANISATION DES MIKROCOMPUTER-SYSTEMS

Dann geht er zum nächsten Befehl im Programm und führt ihn aus. Dieser Vorgang, einen Befehl zu holen und ihn auszuführen wird jedesmal mit dem nächsten Befehl in der Folge ausgeführt.Auf diese Weise leitet das Programm den Prozessor, um die gewünschte Arbeitsweise des Systems zu erreichen.

1.1.3 Begriffe und Konzepte zur Adressierung

Bevor die Gesamtarbeitsweise des Systems genauer besprochen wird, sollen einige Begriffe und Konzepte für die Adressierung erklärt werden. Das erleichtert das Verständnis der nachfolgenden Besprechung.

1.1.3.1 Bit

Bit ist ein allgemeiner Begriff für alles, das einen binären Wert, d.h. einen Wert von 0 oder 1 aufweist. So ist ein 8 Bit-Datenbus ein Satz von 8 Leitungen, denen man einen Wert von logisch o oder logisch 1 zuweisen kann.Die logischen Werte auf diesen Leitungen werden von zwei verschiedenen Spannungen oder Strömen dargestellt. Auf ähnliche Weise kann eine binäre Anzeige für 16 Bit mit 16 einzelnen Lampen aufgebaut werden. Die logische 1 wird durch eine eingeschaltete Lampe angezeigt.

Im folgenden Text ist die Rede von einem 8 Bit-Datenbus, einem 16-Bit-Adressbus, 4 Bit-Daten, 8-Bit-Registern usw. In allen Fällen bleibt die Definition des Begriffes Bit die gleiche.

1.1.3.2 Adressbereich

Zum Verständnis eines Mikrocomputer-Systems ist der Begriff „ Adressbereich " sehr nützlich. Er bezieht sich auf die gesamte Menge an Adressen, die der Mikroprozessor erzeugen kann. Z.B. könnte ein Prozessor mit nur 4 Adressleitungen nur die Adressen 0 - 15 (binär 0000 bis binär 1111) erzeugen.

Das reicht für den Betrieb eines Mikrocomputers nicht aus und deshalb hat der typische Prozessor zwischen 12 und 16 Adressleitungen. Da jede Leitung einen Wert von 0 oder 1 annehmen kann, können diese Bausteine gewöhnlich zwischen 4.096 bis 65.536 verschiedene Adressen ansprechen. Bild 1.2 enthält eine bildliche Darstellung des Adressbereiches für einen typischen 8-Bit Mikrocomputer mit 16 Adressleitungen. Außerdem führt dies Bild das Konzept der PAGE ein.

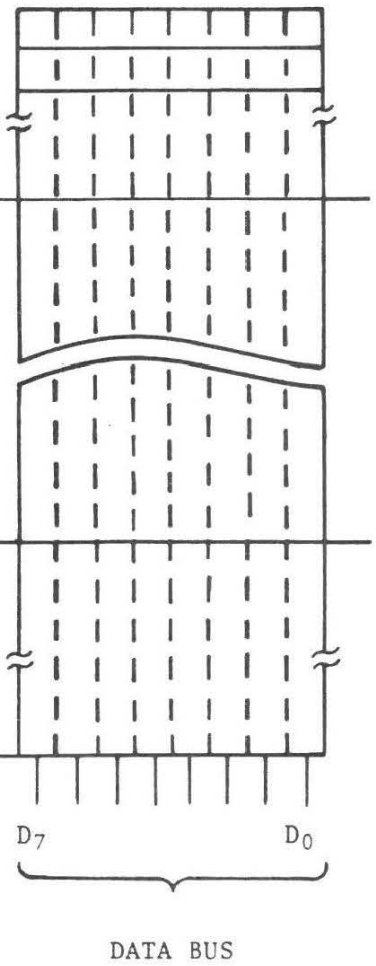
1.1.3.3 Adress-Seite (PAGE)

Das Konzept der PAGE im Speicher ist in 8-Bit Mikrocomputer-Systemen sehr wichtig. Der interne Aufbau eines 8-Bit Prozessors umfaßt 8-Bit-Register, 8-Bit paralle Datenwege usw. Die meißten arithmetischen und logischen Operationen usw. werden mit 8-Bit-Daten zur Zeit vorgenommen. Ebenso ist der 16-Bit-Zähler, der bestimmt welcher Befehl ausgeführt wird, tatsächlich in zwei Busse von 8-Bit geteilt. Einer enthält die Bits 0 - 7 (niederwertige Adressenbits) und der andere die Bits 8 - 15 (höherwertige Adressenbits). Man kann sich also den Adressbereich in Bild 1.2 als aus 256 Blöcken bestehend vorstellen, von denen jeder aus 256 getrennten Adressen besteht. Jeder dieser Blöcke wird als „ PAGE " des Speichers bezeichnet. Die höherwertigen 8-Bits der Adresse (ADH) geben daher die Page an, auf der sich die Adresse befindet, un die niederwertigen 8-Bits (ADL) bezeichnen eine bestimmte Adresse auf dieser page. Die erste Page im Speicher (ADH = 00) wird als Page 0 bezeichnet. Die nächsthöhere Page (ADH = 01) als Page 1 usw.

1.1.4 Bauteile des Systems

Das Blockschaltbild in Bild 1.1 zeigt die Grundbauteile aus denen alle Mikrocomputer-Systeme bestehen. Jeder dieser Blöcke kann aus einer oder mehreren integrierten Schaltungen bestehen oder die Funktionen können auf einzelnen Chips zusammengefaßt sein. Dadurch wird jedoch die grundlegende Arbeitsweise nicht beeinflußt.

BINARY ADDRESS																DECIMAL	HEXADECIMAL Address Code			ADDRESSABLE MEMORY FIELD
High Order								Low Order								WORD	Page Number		Byte Number	(65536 Bytes)
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	NUMBER				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	-	00	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	00	-	01	
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	255	00	-	FF	
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	256	01	-	00	
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	65279	FE	-	FF	
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	65280	FF	-	00	
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	65281	FF	-	01	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	65535	FF	-	FF	



Address Bus and Relation to Memory Field

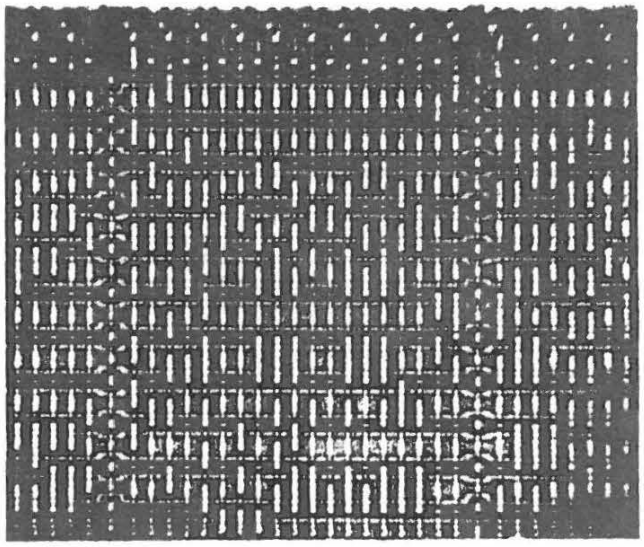
ABBILDUNG 1.2: ADRESSBUS UND BEZEICHNUNG ZUM SPEICHERFELD

1.1.4.1 Taktgenerator

Der Taktgenerator erzeugt eine fortlaufende Kurvenform, die normalerweise zur Steuerung aller Signalübergänge im System benutzt wird. Er ist das „ Herz " des Systems. In einem typischen Mikrocomputer-System ändert sich der Adressbus während der einen Hälfte des Taktzyklus und die Daten werden während der zweiten Hälfte übertragen. Zusätzlich zur Auswertung von Adresse, Daten und Steuerleitungen müssen Prozessor und Unterstützungs-Bausteine auch den Systemtakt überprüfen, um festzustellen, wann sie Daten ausgeben und wann sie Daten abspeichern müssen, die von einem anderen Baustein erzeugt worden sind.

1.1.4.2 Programmspeicher

Der Programmspeicher speichert die Folge von Befehlen, aus denen das Systemprogramm besteht. Wie jeder Speicher legt diese Einheit ein Muster von Einsen und Nullen auf den Datenbus wenn eine Adresse auf den Adressbus-Eingang gegeben wird. Jede einzelne Adresse wählt einen Satz von 8-Bit aus und legt diese Daten auf den Datenbus. Es kommt nicht darauf an wo die Adresse erzeugt worden ist oder wo die Daten benutzt werden. Der Speicher gibt für eine vorgegebene Adresse die entsprechenden 8-Bit-Daten auf den Datenbus. Eine einzigartige Eigenschaft der meisten Mikroprozessor-Systeme ist die Speicherung des Programms in Festwertspeichern. Die Daten sind in einem festen Muster von Bits im Speicher festgelegt. Bild 1.3. zeigt einen Ausschnitt aus einem Halbleiter-Festwertspeicher (ROM).



Portion of Read Only Memory Matrix

ABBILDUNG 1.3: TEIL EINER ROM-MATRIX

Da die Daten im physischen Aufbau des Bausteins gespeichert sind, bleiben sie auch beim Abschalten der Stromversorgung erhalten. Zur Bereitstellung des Systemprogramms ist es nur nötig den Baustein in seinen Sockel zu stecken. Der Ausdruck Festwertspeicher bedeutet, daß der Prozessor die im Baustein gespeicherten Daten beim Systembetrieb nicht ändern kann. Der Prozessor kann nur die Daten auslesen, die im Baustein während des Herstellungsprozesses gespeichert worden sind. Zum Auslesen wird die Adresse an den Baustein gegeben um die entsprechenden 8-Bit-Daten auf dem Datenbus zu haben.

1.1.4.3 Datenspeicher

Für die zeitweise Speicherung von Eingangsdaten, von Ergebnissen arithmetischer Operationen usw. verwendet der Mikrocomputer einen Lese/Schreibspeicher (RAM, Random Access Memory). Der Prozessor kann im RAM Daten speichern (einschreiben) oder er kann die gespeicherten Daten auslesen.

Wie bei dem ROM entspricht jede Adresse 8 Speicherzellen. Die Daten im RAM müssen jedoch vom Prozessor in den Speicher gegeben werden. Sie werden in Flip-Flops gespeichert. Wenn die Stromversorgung zum Baustein abgeschaltet wird, gehen alle dort gespeicherten Daten verloren. Die Daten werden als „ flüchtig " (volatile) bezeichnet. Daten in einem ROM gehen bei Spannungsabschaltung nicht verloren und werden daher als „ nicht-flüchtig" (non-volatile) bezeichnet.

Das Einschreiben der Daten in das RAM findet statt wenn das Signal Write-enable in den Schreibzustand geht. Zu dieser Zeit werden die Daten auf dem Datenbus in den 8 Speicherzellen gespeichert, die der Adresse auf dem Adressbus entsprechen. Der Prozessor kann diese Daten durch Anlegen der richtigen Adresse und das Halten des Signal Write-enable im Lesezustand auslesen.

1.1.4.4 Eingangs/Ausgangs-Bausteine

Die Eingangs/Ausgangs-Bausteine sind die Schaltungen, mit denen Drucker, Tastaturen, Anzeigen usw. mit dem Prozessor verbunden werden. Sie erlauben dem Prozessor Daten von einer Tastatur auszulesen, den Zustand von Fühlern und Schaltern zu prüfen und die Ergebnisse interner Arbeitsgänge anzuzeigen oder zu drucken.

Gleich wo Daten erzeugt werden, sie müssen in der Form von Einsen und Nullen vorliegen, bevor der Prozessor mit ihnen arbeiten kann.Ebenso müssen vom Prozessor eingeleitete Handlungen durch Einsen oder Nullen ausgelöst werden, die vom Prozessor zu Ausgangsleitungen übertragen werden.

Die Übertragung von Daten aus dem Prozessor zu einem Ausgangs-Baustein wird normalerweise durch Schreiben der Daten erreicht. Das geschieht fast in der gleichen Weise wie beim Einschreiben in ein RAM. Jeder Satz von 8 Eingangs- oder Ausgangs-Leitungen (als Kanal bezeichnet) bekommt eine Adresse und der Prozessor schreibt einfach Daten in diese Adresse.

Für jede in einen peripheren Kanal eingeschriebene 1 wird ein Ausgang hochgesetzt und für jede 0 tief.

Das Konzept der peripheren Steuerung ist einfach. Die tatsächliche Realisierung dieser Schnittstellen kann jedoch viele komplizierte Verfahren beinhalten, um die Fähigkeit des Prozessors Peripherie-Geräte zu steuern und gleichzeitig interne Arbeitsgänge auszuführen zu vergrößern. Diese Verfahren werden in Kapitel 2 dieses Handbuchs im einzelnen besprochen.

1.1.4.5 Der Mikroprozessor

Auf den ersten Blick erscheint es seltsam die Unterstützungs-Bausteine in einem Mikroprozessor-System vor Erwähnung des Prozessors zu besprechen. Dies Vorgehen wird jedoch nötig durch die Tatsache, das die meisten Ein- und Ausgänge zur richtigen Steuerung der Unterstützungs-Bausteine und Peripheriegeräte dienen.

Der Adressbus, der Datenbus für zwei Richtungen und die Leitung Write-Enable erlauben dem Prozessor die direkte Steuerung des Systems. Der Adressbus gibt die Daten aus, die die Quelle oder den Bestimmungsort von Datenübertragungen steuern. Diese Adressen werden von verschiedenen Quellen innerhalb des Prozessors abgeleitet. Zum Holen von Befehlen aus dem Programmspeicher werden die Adressen üblicherweise von einem Zähler abgeleitet, der die Ausführung aufeinanderfolgender Befehle steuert. Adressen für die Datenübertragung zwischen Prozessor und RAM werden normalerweise direkt vom Programm abgeleitet oder aus Daten im Programm und Daten in internen Registern gerechnet.

Der Datenbus in zwei Richtungen dient als Weg zur Obertragung von Daten in den und aus dem Prozessor. Die Richtung der Datenübertragung wird durch die Leitung Write-Enable bestimmt.

Eine weitere besondere Funktion in modernen Mikrocomputer-Systemen ist der Interrupt. Mit dieser Funktion können die Peripherie-Geräte die Arbeitsweise des Prozessors direkt beeinflussen. Sobald ein Interrupt-Signal erzeugt wird, führt der Prozessor seinen gerade laufenden Befehl zu Ende aus und antwortet dann unter Programmsteuerung auf den Interrupt. Auf diese Weise kann der Prozessor das Systemprogramm durchführen, ohne daß das Systemprogramm den Zustand des Peripherie-Gerätes zu überwachen braucht. Die Software zum Betrieb jedes Peripherie-Gerätes wird nur dann ausgeführt, wenn sie nötig ist.

1.2 EINFÜHRUNG IN DIE MIKROPROZESSOR-FAMILIE MCS 650X

Das Angebot der MOS Technology, INC. an Mirkoprozessoren besteht aus dem MCS 6502, einem Baustein mit 40 Stiften, und drei Prozessoren mit 28 Stiften, den MCS 6503, MCS 6504 und MCS 6505. Jeder dieser Bausteine zielt auf einen bestimmten Anwendungsbereich. Daher soll ein genauer Überblick über ihre Fähigkeiten und die Unterschiede zwischen ihnen gegeben werden.

Der MCS 6502 soll in allen Entwicklungen Verwendung finden, die einen vollen 16 Bit Adress-Bus erfordern. Die Prozessoren mit 28 Stiften können jedoch in kleinen kostenempfindlichen Systemen Einsparungen in Prozessorkosten und Leiterplattenfläche bringen. Der MCS 6503, MCS 6504 und MCS 6505 finden Anwendung in allen Entwicklungen, in denen das System innerhalb ihrer Adressier-Grenzen arbeitet.

1.2.1 Der MCS 6502

Das erste Mitglied der Prozessor-Familie ist ein Baustein mit 40 Stiften. Er sollte in allen Entwicklungen verwendet werden, in denen die Möglichkeiten eines Prozessors mit 40 Stiften nötig sind. Taktoszillator und Takttreiber befinden sich im Baustein. Die Takttreiber können mit einem einzigen TTL-Rechtecksignal getrieben werden oder mit dem internen Oszillator. Die Schwingfrequenz des Oszillators kann durch Anschluß einer RC-Kombination festgelegt werden. Für größere Stabilität der Taktfrequenz kann auch ein Quarz benutzt werden.

Der MCS 6502 verfügt über einen vollen 16 Bit Adress-Bus, einen Daten-Bus für 8 Bit in zwei Richtungen und zwei Interrupts. Außerdem zeigt er durch ein Synchronisations-Signal die Zyklen an, in denen der Prozessor einen Operations-Code vom Programmspeicher holt.

1.2.2 Der MCS 6503, MCS 6504 und MCS 6505

Es gibt drei Versionen des Prozessors mit 28 Stiften. Sie unterscheiden sich in der Zahl der Adressleitungen und der Interrupts. Dadurch kann der Entwickler den Prozessor wählen, der für seine besondere Anwendung am besten geeignet ist.

Der MCS 6504 kann mit seinen 16 Adressleitungen 8 K Byte adressieren. Er hat jedoch nur einen Eingang zur Anforderung eines Interrupts, $\overline{IRQ}$. Der nicht maskierbare Interrupt, $\overline{NMI}$ ist nicht herausgeführt.

Der MCS 6503 und der MCS 6505 haben eine Adressleitung weniger. Beim MCS 6503 wird diese Adressleitung durch einen zweiten Interrupt-Eingang, $\overline{NMI}$, ersetzt. Beim MCS 6505 wird diese Adressleitung durch das Signal RDY ersetzt. Alle anderen Funktionen sind bei diesen Prozessoren die gleichen. Die Einzelheiten jedes dieser Anschlüsse sind in den folgenden Abschnitten besprochen.

Die Arbeitsweise der verschiedenen Busse, Steuersignale usw. ist bei allen MCS 650X - Erzeugnissen genau die gleiche. Alle Prozessoren entsprechen den Systemdaten, die in Abschnitt 1.3. dieses Handbuchs besprochen werden.

1.3 MCS 6500 SYSTEM KONZEPTE

1.3.1 Bus Struktur

Das Mikrocomputer System MCS 6500 ist um zwei Hauptbusse organisiert. Jeder Bus besteht aus einem Satz paralleler Leitungen, die zur Obertragung binärer Information zwischen den Bausteinen des Systems benutzt werden können. Der erste Bus, der Adressbus, überträgt die Adresse die der Prozessor erzeugt, auf die Eingänge des Speichers und der Peripherie Interface Bausteine. Da der Prozessor die einzige Quelle von Adresse in einem normalen System ist, wird dieser Bus als Einrichtungsbus bezeichnet. Beim MCS 6502 besteht der Adressbus aus 16 Leitungen. Damit kann der Prozessor auf insgesamt 65.536 Speicherworte, Register usw. zugreifen. Der Adressbus der MCS 6503, MCS 6504 und MCS 6505 umfaßt weniger Leitungen. Deswegen arbeiten sie mit einem kleineren „Adressbereich". In Abschnitt 1.1.3 wird das im einzelnen besprochen.

Der Datenbus im Mikrocomputer System MCS 6500 besteht aus einem 8 Bit Datenweg in zwei Richtungen. Während einer Schreiboperation (WRITE) übertragen diese Leitungen Daten vom Prozessor zum ausgewählten Speicherwort, usw. und während einer Leseoperation (READ) vom Speicher in den Prozessor. Alle Daten und alle Befehle werden auf den Datenbus übertragen.

Die READ/WRITE (R/W) Leitung des Prozessors steuert die Richtung der Datenübertragung. Diese Leitung führt die Funktion Write Enable aus, die in Abschnitt 1.1.4.3 beschrieben ist. Solange die R/W Leitung hochliegt (größer als 2,4 Volt DC), gehen alle Datenübetragungen vom Speicher zum Prozessor (READ). Diese Leitung geht nur auf 0 wenn der Prozessor Daten in den Speicher einschreiben will.

Wie in den meisten Mikrocomputer Systemen steuert der Systemtakt den Zeitablauf aller Datenübertragungen. Der Takt selbst besteht aus zwei nicht überlappenden Rechteckwellen. Dieses Zwei-Phasen-Taktsystem kann man sich am besten als zwei abwechselnd positiv gehende Pulse vorstellen. Im Text werden wir auf den Takt als Phase 1 und Phase 2 bezugnehmen. Im Besonderen stabilisieren sich Adressleitungen und die R/W Leitung während der Phase 1, und alle Datenübertragungen finden während Phase 2 statt.

In Bild 1.5 werden die Zeitbedingungen für den Betrieb bei 1 MHz Taktfrequenz angegeben. Beachten Sie daß die Reihenfolge der Operationen für alle Prozessoren die gleiche ist. Diese Teitfestlegungen ändern sich jedoch für Bausteine die schneller als 1 MHz arbeiten. Die Adresse steht 300 Nanosekunden nach der steigenden Flanke von Phase 1 stabil an und die Daten müssen 100 Nanosekunden vor der fallenden Flanke von Phase 2 stabil anstehen. Bei 1MHz haben die Speicherbausteine etwa 575 Nanosekunden Zeit, Daten auf den Bus zu geben. Der grundlegende Ablauf der in Abb. 1.6, 1.7 und 1.8 gezeigt wird ändert sich nicht obwohl es viele Faktoren gibt, die die tatsächlichen Daten und Adressen bestimmen, die im System erzeugt werden. Diese Abbildungen legen das Verfahren auf dem Systembus fest, das für alle MOS Technology, Inc. Prozessoren und Unterstützungsbausteine gilt.

1.3.2 Prozessor Interrupts

Die peripheren Geräte (Drucker, Tastaturen, usw.) können durch Erzeugen von Prozessor Interrupt Signalen Bedienung anfordern. Obwohl dies Verfahren im Konzept recht einfach ist, stellt die richtige Erzeugung und Steuerung der Interrupts eines der wichtigsten Probleme dar, die sich dem Entwickler stellen. Die Fähigkeiten des gesamten Systems können stark ausgeweitet werden wenn der Prozessor nur dann Software für die Peripherie ausführen muß, wenn es unbedingt nötig ist.

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
Cycle Time	T_{CYC}	1.0 µs	--	--	µsec
Clock Pulse Width Ø1 (Measured at Vcc-0.2v) Ø2	PWH Ø1 PWH Ø2	430 430	--	--	nsec
Rise and Fall Times (Measured from 0.2V to Vcc-0.2V)	T_F, T_R	--	--	25	nsec
Delay time between Clocks (Measured at 0.2V)	T_D	0	--	--	nsec

CHARACTERISTIC	SYMBOL	MIN.	TYP.	MAX.	UNIT
Read/Write Setup Time from MCS650X	T_{RWS}	--	100	300	ns
Address Setup Time from MCS650X	T_{ADS}	--	200	300	ns
Memory Read Access Time T_R $T_{CYC} - (T_{ADS} - T_{DSU} - tr)$	T_{ACC}	--	--	500	ns
Data Stability Time Period	T_{DSU}	100	--	--	ns
Data Hold Time	T_H	10	30	--	ns
Enable High Time for DBE Input	T_{EH}	430	--	--	ns
Data Setup Time from MCS650X	T_{MDS}	·	150	200	ns

Clock and Read/Write Timing Table (1MHz Operation)

ABBILDUNG 1.5: ZEIT-TABELLE FÜR TAKT UND SCHREIBEN/LESEN

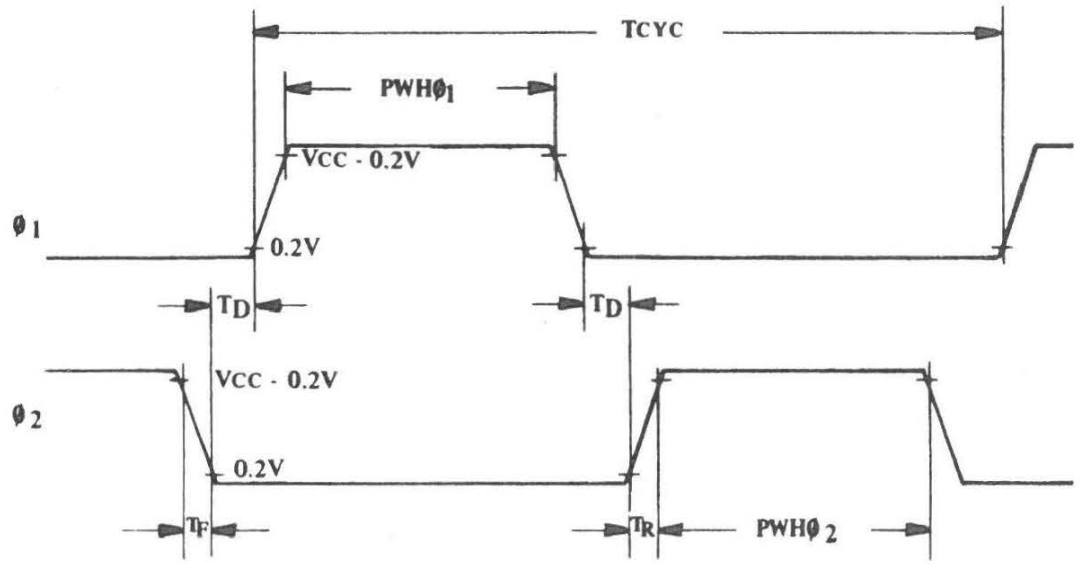
Da das ganze System auf die richtige Weise auf die Interrupt Eingänge reagieren muß, ist die Interrupt Struktur hauptsächlich ein Problem der Kompliziertheit des Systems. Die tatsächlichen Signale auf die das System reagieren muß liegen üblicherweise an den Eingängen eines Peripherie Interfacebausteins an. Die Interrupts werden in diesem Baustein angeschaltet (enabled) abgeschaltet (disabled) und abgespeichert (latched) bis der Interrupt abgearbeitet wird. Der Peripherie Interfacebaustein erzeugt Signale die den Anforderungen der Prozessor Interrupt Eingänge entsprechen.

Es gibt zwei Interrupt Leitungen in den Mikroprozessor, $\overline{\text{IRQ}}$ (Interrupt Request) und $\overline{\text{NMI}}$ (Non-Maskable Interrupt).

Da die Bedingungen der beiden Interrupt Eingänge verschieden sind, werden sie unten getrennt besprochen. Die Antwort des Prozessors auf diese Eingänge ist jedoch sehr ähnlich nachdem der Interrupt erkannt worden ist. Aus diesem Grund wird die innere Wirkungsweise des Prozessors während der Bedienung eines Interrupts in der genauen Analyse des Prozessorchips besprochen. Stattdessen liegt in diesem Abschnitt das Hauptgewicht auf den Überlegungen auf Systemebene, die nötig sind, um die richtige Arbeitsweise der Interrupt Struktur zu gewährleisten.

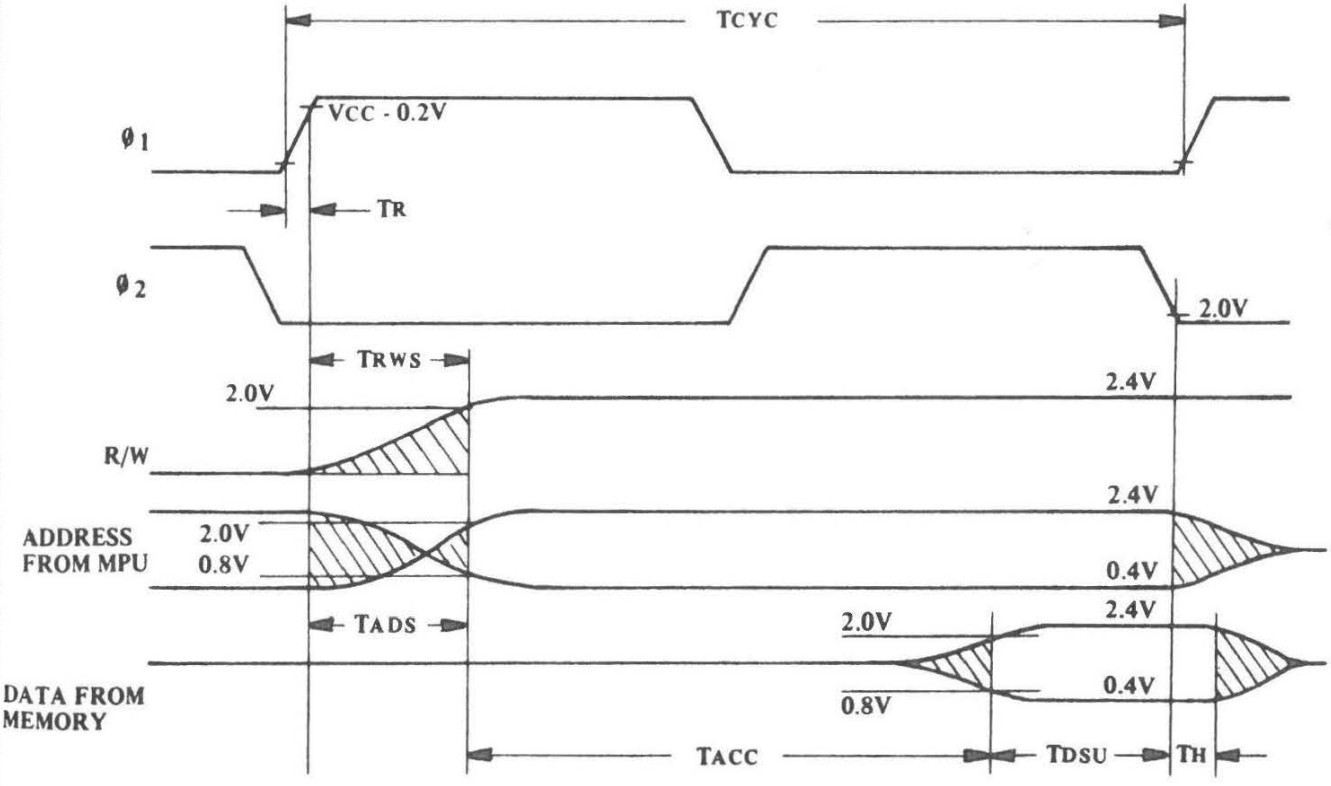
1.3.2.1 Anwendungen für Interrupts

Die Bestimmung derjenigen Signale, die Prozessor Interrupts hervorrufen und derjenigen Operationen, die als Antwort auf diese Interrupts stattfinden ist eine der wichtigsten Aufgaben die sich dem Mikrocomputer Systementwickler stellen. Eine ins einzelne gehende Besprechung dieser Überlegungen ist in Kapitel 2 dieses Handbuches enthalten. Hier sollen jedoch einige Beispiele von interruptgetriebenen Operationen vorgestellt werden, um dem Entwickler ein Gefühl dafür zu geben, warum dies Verfahren in Mikrocomputersystemen so häufig benutzt wird.



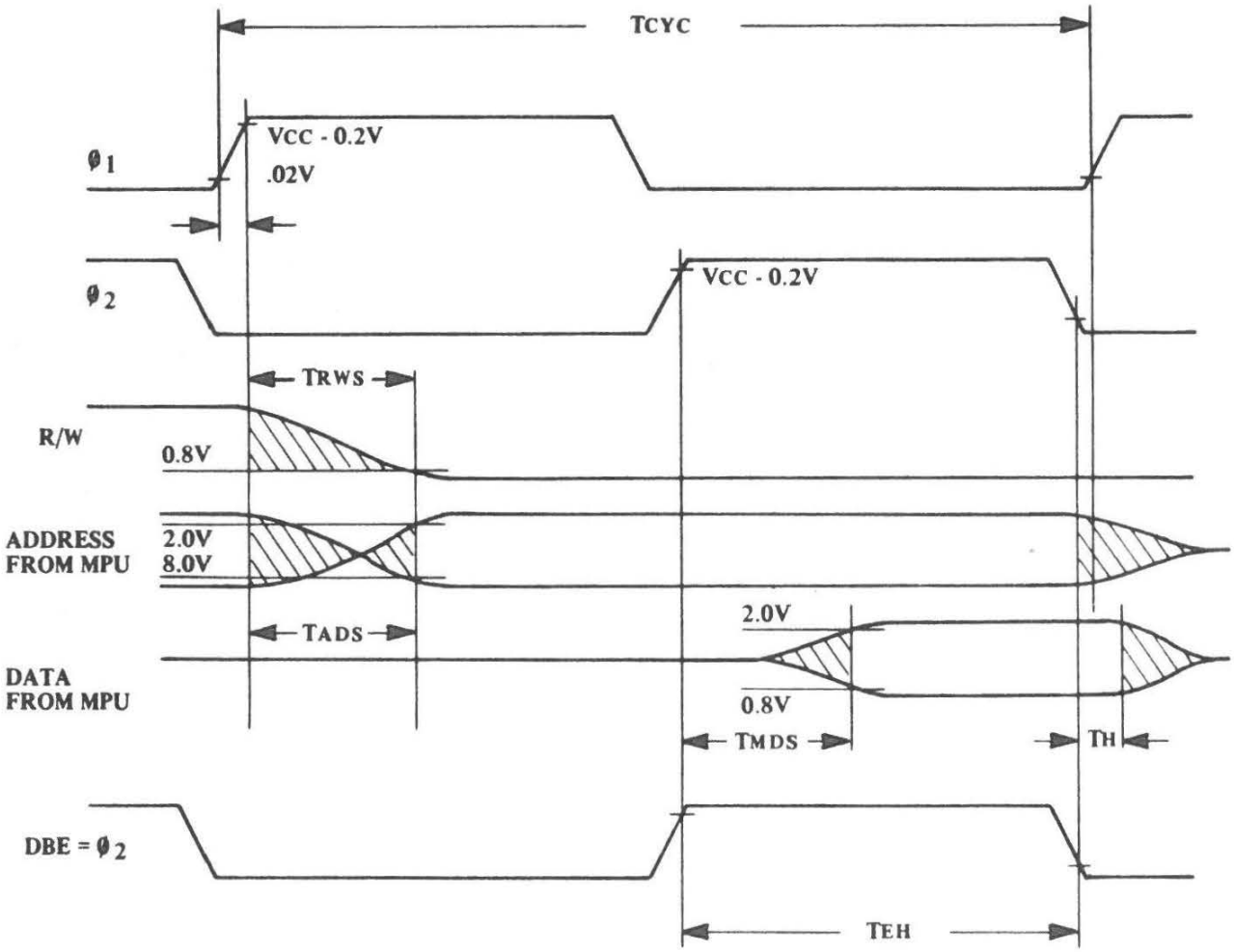
Two Phase Clock Timing

ABBILDUNG 1.6: ZEITDIAGRAMM DES ZWEIPHASEN-TAKTES



Timing for Reading Data from Memory or Peripherals

ABBILDUNG 1.7: ZEITDIAGRAMM FÜR LESEN AUS SPEICHER ODER PERIPHERIE



Timing for Writing Data to Memory or Peripherals

ABBILDUNG 1.8: ZEITDIAGRAMM FÜR SCHREIBEN IN SPEICHER ODER PERIPHERIE

Beispiel 1 Eine vollständig decodierte Tastatur

In vielen Systemen wird das Problem der Dateneingabe durch eine Tastatur gelöst. In kleinen Systemen kann der Prozessor die Auswertung der binären Codes übernehmen, die jeder Taste zugeordnet sind. In großen Datenterminals enthält jedoch die Tastatur üblicherweise einen Encoder, der für jede Taste einen bestimmten Code erzeugt. Wenn eine Taste gedrückt wird, wird der entsprechende Code an die Ausgangsstifte gelegt und als Strobesignal erzeugt, das anzeigt, daß eine Taste gedrückt worden ist.

Die Tastatur stellt eine ideale Anwendung für interrupt getriebenen Betrieb dar. Die Interrupts geschehen selten und die auszuführende Operation ist recht einfach. Die Strobeleitung der Tastatur wird direkt mit einem Interrupteingang an einen Peripherie Interface Baustein verbunden. Jedes Mal, wenn ein Strobesignal erzeugt wird, wird ein Interrupt ausgelöst und der Prozessor liest die Daten im Peripherie-Kanal in den Speicher, analysiert diese Daten und kehrt dann zu dem Programm zurück, das gerade bearbeitet wurde. Wenn keine Taste gedrückt wird, verliert der Prozessor keine Zeit in der Bedienung der Tastatur.

Ohne Interrupts müßte der Prozessor regelmäßig die Tastaturdaten auslesen um eine betätigte Taste zu erkennen. Diese Operation müßte ungefähr alle 50-100 Millisekunden geschehen. Zusätzlich zur Erkennung einer gedrückten Taste müßte der Prozessor feststellen, daß jeder einzelne Tastendruck einmal und nur einmal erkannt wird. Das wird in Abschnitt 1.3.2.5 und 1.3.2.6 besprochen. Diese Software ist viel komplizierter als die einfache Interrupt Routine. Ein anderer Nachteil der Verarbeitung ohne Interrupt ist, daß der Prozessor regelmäßig einen Teil der Zeit für das Abfragen verwendet. In vielen Systemen ist das kein Problem, aber in großen Terminals, usw. könnte die Zeit für das Überprüfen der Tastatur-Strobes besser für andere Operationen verwendet werden.

Darum muß der Entwickler sich fragen, ob der Prozessor im zu entwickelnden System die Überprüfung des Tastatur-Strobe ausführen kann und außerdem seine anderen Aufgaben bewältigen.

Beispiel 2 - Eine abgetastete Anzeige

Obwohl Zeit bei der Bestimmung der Notwendigkeit von Interrupts ein bedeutender Faktor ist, kann das Interrupt-Verfahren auch bei parallelen Operationen sehr nützlich sein. Ein hervorragendes Beispiel dafür gibt ein System das eine digitale Anzeige und/oder einen Drucker enthält.

Eine digitale Anzeige wird üblicherweise abgetastet. Dabei wird jedes Digit für eine kurze Zeit in der Folge getrieben. Die ganze Anzeige wird mit einer Frequenz abgetastet, die das Auge nicht wahrnimmt. Die Anzeige braucht in festen Abständen die Aufmerksamkeit des Prozessors. Für den Prozessor ist es sehr schwierig sich wiederholende Zeitabschnitte zu berechnen, während er sein normale System-Programm-Routinen ausführt. Der Prozessor zöge es vor, das Systemprogramm ohne Berücksichtigung der Zeitabstände für die Anzeige laufen zu lassen, und die Display-Software nur dann auszuführen wenn sie benötigt wird.

Eine Lösung für das obige Problem ist die Erzeugung von Prozessor-Interrupts in festen Zeitabständen durch einen externen Zähler oder Takt. Jedes Mal wenn ein Interrupt vorliegt werden die Daten für das nächste Digit im Display an einen Ausgangskanal gelegt. Der Prozessor kehrt dann zu dem Programm zurück, das er vorher ausgeführt hatte.

Beide oben beschriebenen Operationen stellen Lösungen für Systemprobleme dar. Ereignisse die sehr unregelmäßig geschehen und Ereignisse die parallel mit anderen Ereignissen oder parallel mit

dem Hauptsystemprogramm bearbeitet werden müssen sollten ernsthaft als Möglichkeiten für Interrupts in Betracht gezogen werden. Zusätzliche Erwägungen sind in Kapitel 2 dieses Handbuches beschrieben. Das typische System kann mehrere Quellen für Interrupts haben, jede mit ihrer eigenen Zeitfolge und jede mit ihrem eigenen Satz an Operationen, die als Antwort auf die Interrupts ausgeführt werden müssen.

1.3.2.2 Setzen der Interrupt Prioritäten

Nach sorgfältiger Analyse des Gesamtsystems der Bestimmung aller Quellen für Interrupts stellt sich die Frage: „Was geschieht wenn mehr als eine Interrupt Quelle zur selben Zeit bedient werden muß ?" Zwischen den verschiedenen Interrupt Quellen müssen Prioritätsebenen festgelegt werden. Welche müssen innerhalb sehr kurzer Zeit beachtet werden? Welche können eine Zeit aufgeschoben werden? Diese Zuweisung von Prioritäten und das Verfahren aus mehreren gleichzeitigen Interrupts auszuwählen ist für den Betrieb des Systems sehr wichtig und sollte möglichst früh in der Systementwicklung festgelegt werden.

Ein MCS 650X System kann verschiedene Hardwaremethoden zur Bestimmung des Interrupts mit der höchsten Priorität verwenden. Diese beinhalten normalerweise die Verwendung eines „Priority Encoders" der dem Prozessor erlaubt direkt in die Software zu gehen die den Interrupt mit der höchsten Priorität bedient. Nachdem dieser beendet ist geht er auf die nächst höhere Priorität und führt die betreffende Software aus. Die MCS 650X Familie sieht jedoch eine viel weniger aufwendigere Methode der Zuweisung von Interrupt Prioritäten vor. Das ist der „abgefragte" Interrupt. Jedesmal wenn eine aktive Interruptquelle erkannt wird führt der Prozessor bei diesem Verfahren ein „abgefragtes" Interrupt Programm durch, das den Interrupt mit der höchsten Priorität abfragt, dann den nächsthöchsten usw. bis ein aktiver Interrupt festgestellt wird. Das Programm bedient diesen Interrupt und kehrt zu dem „abgefragten" Interrupt Programm zurück.

Es fährt mit der Abfrage des Interrupts mit der nächsthöheren Priorität fort bis alle abgefragt sind oder es löscht den Interrupt desable um verschachtelte Interrupts zuzulassen. Das Programm für den „abgefragten" Interrupt wird immer ausgeführt wenn ein Interrupt auftritt. Dadurch werden alle Interrupts die gleichzeitig auftreten nacheinander in Reihenfolge der Priorität bedient.

In Kapitel 2 dieses Handbuchs werden mehrere Hardware Verfahren für die Zuweisung von Interrupt Prioritäten besprochen. Im nächsten Abschnitt wird die Systemverbindung beschrieben, die die Verwendung des einfachen „abgefragten" Interrupt ermöglicht.

Der Zugriff zu den verschiedenen Bits in den Steuerregistern erfolgt während eines Programmes viele Male, damit der Prozessor Interrupts ein- oder ausschalten, Betriebsarten ändern usw. kann, wie es das gesteuerte periphere Gerät erfordert.

1.3.2.3 System-Verbindung für Interrupts

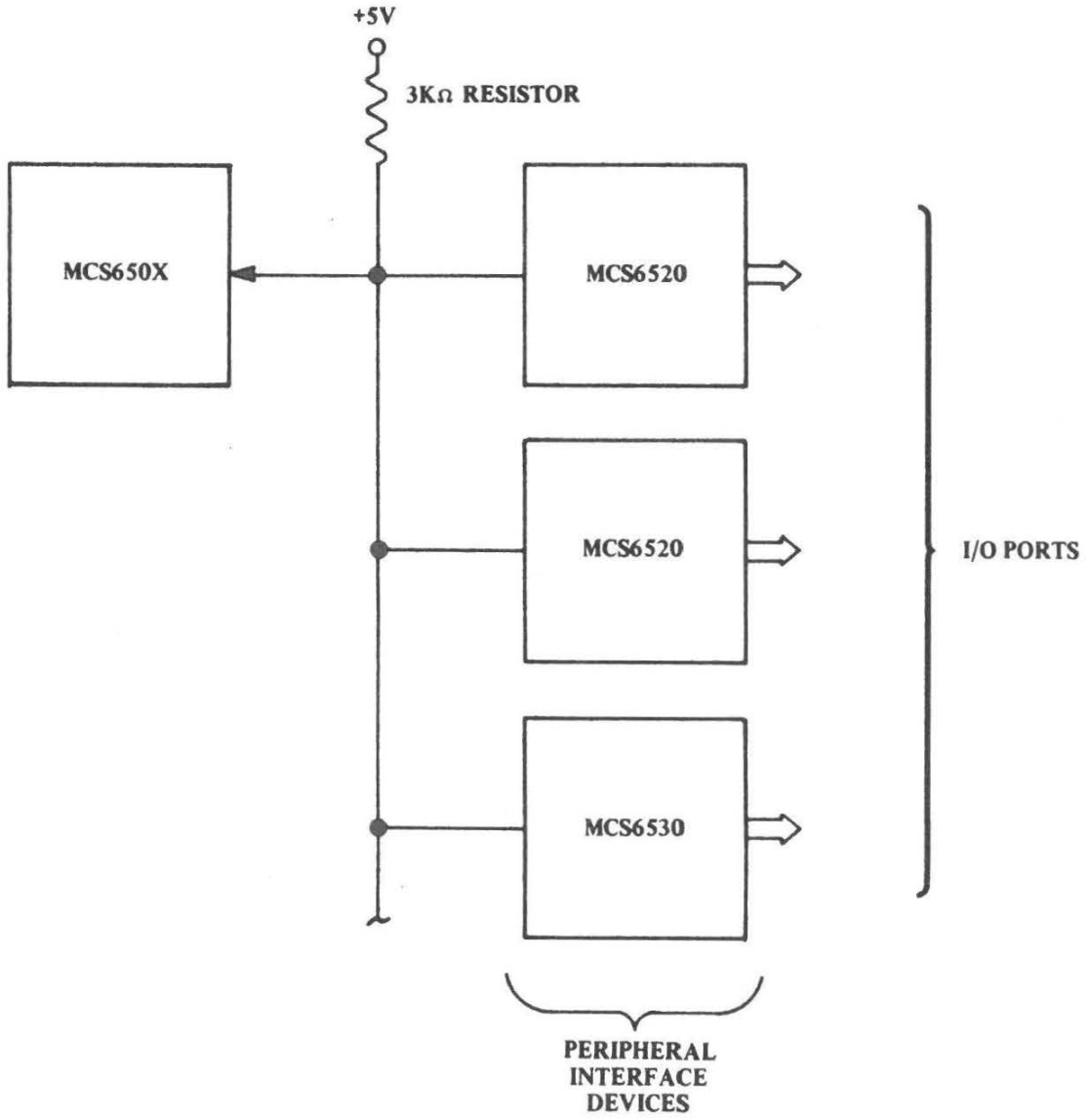
Beim einfachen "abgefragten" Interrupt-Verfahren bestimmt die Interrupt-Software den aktiven Interrupts mit der höchsten Priorität. Die Anforderungssignale für den Interrupt, $\overline{IRQ}$ oder $\overline{NMI}$, veranlassen den Sprung des Prozessors zur Software für die Anfrage.

Die verschiedenen Interrupt -Signale können deswegen mit einem logischen ODER verbunden werden, um das Signal für den Prozessor zu erzeugen. Jeder aktive Interrupt veranlasst dann den Prozessor zur Abfrage und Bedienung. In den peripheren Schnittstellen-Bausteinen der Familie ist die Erzeugung dieses logischen ODER vorgesehen. Da diese Peripherie-Adapter viele der Einschalt- und Speicherfunktionen ausführen, die für die richtige Bedienung des Interrupt nötig sind, ergibt ihr Ausgang das Signal, das den Interrupt im Prozessor auslöst. Diese Interruptausgänge können mit einem " verdrahteten - ODER -" zusammengeschaltet werden. Diese einzelne Leitung wird mit einem Widerstand gegen + 5 V hochgezogen und mit dem Prozessor verbunden. Jeder der Interrupt-Ausgänge in den Peripherie-Adaptern kann dann diesen Interrupt nach unten ziehen. Diese einfache Schaltung wird in Bild 1.9 gezeigt.

1.3.2.4 Bedienung der Interrupts

Im Vorhergehenden ist viel über die Festlegung von Interrupts gesprochen worden und was als Antwort auf einen Interrupt geschieht. Jetzt wird zum besseren Verständnis der Einzelheiten der Interrupt-Eingänge am Prozessor die Folge besprochen, die abläuft, wenn ein Interrupt vom Prozessor erkannt wird.

Eine Interrupt-Anforderung wird durch L am Interrupt-Abfrageeingang angezeigt. Dieser Interrupt wird erkannt, nachdem der Prozessor den gerade laufenden Befehl ausgeführt hat. Im nächsten Schritt wird genug von dem Inhalt der internen Prozessor-Register abgespeichert, daß der Prozessor die Ausführung des Programms, das



Interrupt Wire OR'd Hardware Configuration
from Peripheral Interface Devices to Microprocessor

ABBILDUNG 1.9: INTERRUPTS WIRED -OR- HARDWARE-KONFIGURATION VON PERIPHERIE-SCHNITTSTELLE

unterbrochen worden ist, wieder aufnehmen kann. Im besonderen werden Programmzähler und Prozessor-Status-Register in einer Reihe Speicherstellen abgespeichert, die durch ein anderes internes Register, den Stapelzeiger festgelegt sind. Das Speichern der Inhalte von Programmzähler und Prozessor-Status-Register liegt im Speicher eindeutig fest, wie in Kapitel 9 des Programmier-Handbuches erklärt, in welchem Zustand der Prozessor zur Zeit des Interrupts war. Der Prozessor geht dann an zwei festgelegte Speicherstellen, um den oberen und unteren Teil der Adresse der Software zu bestimmen.

Bis zu diesem Punkt ist der Ablauf automatisch und durch die interne Logik des Prozessors bestimmt. Die Ausführung des Interrupt-Programms beginnt, nachdem der Prozessor die Adresse auf den Adressbus gelegt hat. Alles Nachfolgende wird durch die System-Software bestimmt.

Die gesamte oben beschriebene Interrupt-Software besteht aus einer komplizierten Kombination von Routinen zur Abfragung und Interrupt-Bedienung. Wenn kein Hardware-Verfahren zur Festlegung der Priorität benutzt wird, werden die Verbindungen im System nicht komplizierter als in Bild 1.9 gezeigt.

1.3.2.5 Interrupt-Anforderung ($\overline{IRQ}$)

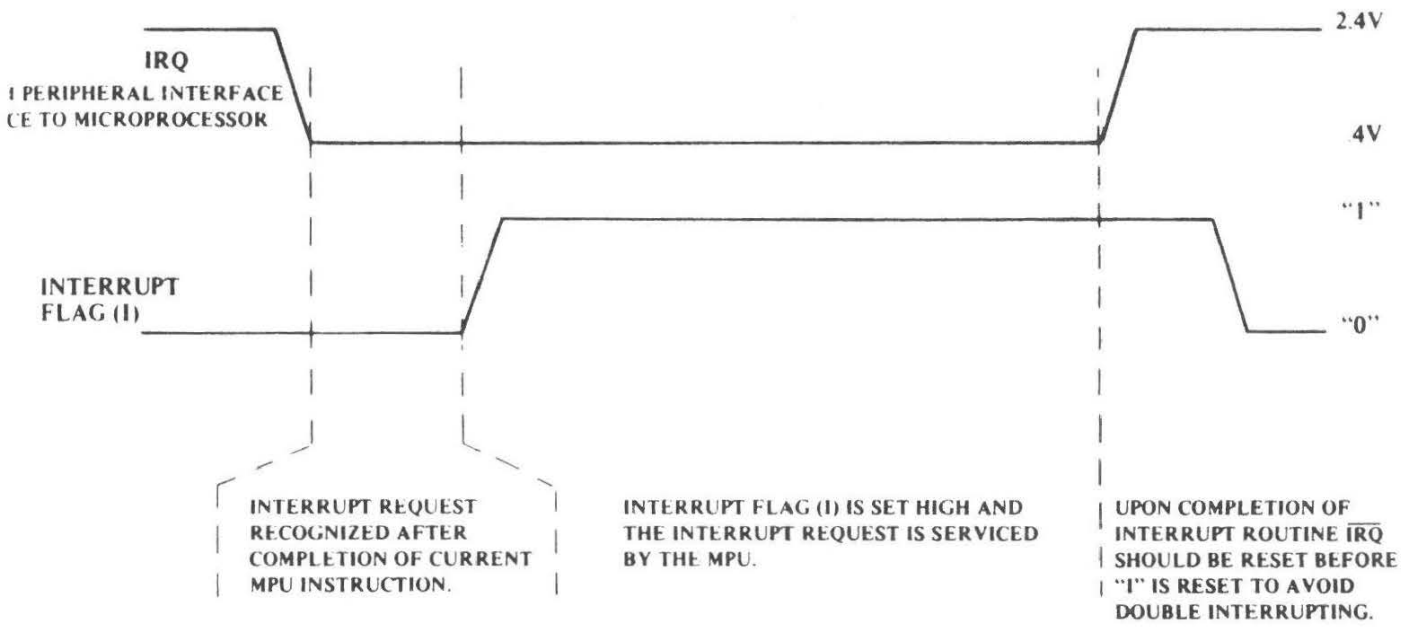
Es gibt, wie in Abschnitt 1.3.2. festgestellt, zwei Interrupt-Leitungen für den Prozessor, $\overline{IRQ}$ und $\overline{NMI}$. Die Anforderungen für die richtige Arbeitsweise des Eingangs für den maskierbare Interruptanforderung ($\overline{IRQ}$) sind strenger als für den zweiten Interrupt-Eingang $\overline{NMI}$. Das liegt hauptsächlich an der Tatsache, daß $\overline{NMI}$ flankengesteuert ist. Beim Eingang $\overline{IRQ}$ bekommt der Prozessor einen Interrupt wenn das Signal am Eingang auf L liegt und das interne Flip-Flop zur Abschaltung des Interrupt rückgesetzt ist.

Dieses Flip-Flop (I) ist ein einzelnes Bit im internen Status-Register des Prozessors. Die Einzelheiten dieses Registers sind in Abschnitt 3.2 des Programmier-Handbuches beschrieben.

Bei der Verarbeitung der Interrupt-Anforderung vom Eingang $\overline{IRQ}$ ist das I-Flip-Flop sehr wichtig. Dieses Teil stellt sicher, daß der Interrupt erkannt und nur einmal für jede Anforderung bedient wird, und nur dann wenn ein Interrupt gewünscht wird. Dies wird im einzelnen unten beschrieben.

Bild 1.10 zeigt den Ablauf während der Bedienung eines $\overline{IRQ}$-Interrupts. Eine positive oder negative Flanke des Signals vom Peripherie-Gerät (Drucker, Tastatur, usw.) wird an den flankenempfindlichen Eingängen des Peripherie-Bausteins erkannt. Wenn der Interrupt im peripheren Schnittstellen-Baustein eingeschaltet ist, geht der Ausgang zur Interrupt-Anforderung ($\overline{IRQ}$) dieses Bausteins auf L. Der Interrupt-Zustand wird im Schnittstellen-Baustein gespeichert, um dem Prozessor ausreichend Zeit zur Abfrage der Interrupt-Quellen zu geben. Damit wird sichergestellt, daß das Interrupt-Signal nicht gelöscht wird, bevor die Abfrage beendet ist. Dies Flip-Flop wird vom Prozessor zurückgesetzt, wenn er die mit diesem Interrupt verbundene Software ausführt. Einzelheiten dieses Vorganges sind in Abschnitt 1.4.1.2.10 beschrieben.

Das I-Flip-Flop (Interrupt-Abschaltung) wird automatisch gesetzt sobald der Prozessor einen Interrupt erkennt. Dadurch wird die erneute Erkennung dieses Interrupts verhindert. Dieses Flip-Flop kann von Hand mit einem Befehl im Programm zurückgesetzt werden oder automatisch mit dem Befehl Rückkehr zum Interrupt (Return from Interrupt). Das I-Flip-Flop darf nicht gelöscht werden, bevor der Interrupt-Eingang zurückgesetzt ist. Wenn der Befehl "I löschen" (Clear I) zu früh im Programm ausgeführt wird kann dieser Interrupt noch einmal erkannt werden. Er wird dann vom Prozessor wie ein neuer Interrupt behandelt.



Sequence to Service $\overline{IRQ}$

ABBILDUNG 1.10: SEQUENZ ZUR BEDIENUNG VON $\overline{IRQ}$

1.3.2.6 Nichtmaskierbarer Interrupt ($\overline{NMI}$)

Der Eingang $\overline{NMI}$ des Prozessors ist flankengesteuert. Zur Auslösung eines Interrupt muß am Eingang $\overline{NMI}$ eine negative Flanke auftreten. Sie löst einen einzelnen Interrupt aus. Nach Bedienung des Interrupts beachtet der Prozessor diesen Eingang so lange nicht bis das Signal auf H und dann zurück auf L geht.

Die Antwort auf das Interrupt-Signal $\overline{NMI}$ kann innerhalb des Prozessors nicht abgeschaltet werden. Nach Vollendung des gerade ausgeführten Befehls erkennt der Prozessor den Interrupt und bedient ihn wie oben beschrieben. Alle Interrupt-Signale sollten gespeichert werden, bis der Prozessor diesen Interrupt bedient hat. Wenn alle Interrupt-Quellen mit den Interrupt-Eingängen von peripheren Schnittstellen-Bausteinen der Familie verbunden sind, ist diese Arbeitsweise sichergestellt.

Die Verarbeitung mehrfacher Interrupts in einer Struktur mit abgefragtem Interrupt erfordert die Abfrage aller Interrupts vor der Ausführung des Befehls " Rückkehr vom Interrupt " (Return from Interrrupt). Das ist wegen des "verdrahteten ODER" zur Zusammenfassung der Interrupts nötig, da nicht bekannt ist welche der Leitungen auf L ging. Wenn einer der Interrupts unbedient bleibt, hält er weiterhin das Signal $\overline{NMI}$ auf L und schaltet damit die Interrupts von allen anderen Quellen ab, da das Signal $\overline{NMI}$ auf H und dann zurück auf L gehen muß, um einen Interrupt auszulösen. Das gilt nicht für den Eingang $\overline{IRQ}$, da dessen Speicher regelempfindlich ist. Der Befehl "Rückkehr vom Interrupt" vor der Bedienung aller $\overline{IRQ}$-Interrupt-Quellen bewirkt nur, daß ein weiterer $\overline{IRQ}$-Interrupt ausgelöst wird.

1.3.3 Rücksetzen des Systems (System Reset)

Das Signal RESET ist eine der grundlegenden Steuerfunktionen des Systems. Ob dieses Signal automatisch durch eine externe Schaltung beim Einschalten oder von Hand durch einen Druckknopf aufgelöst wird, die Teile des Systems müssen dabei festen Regeln gehorchen, damit das System richtig arbeitet. Das gilt im Besonderen für die peripheren Schnittstellen-Bausteine.

In Systemen mit MCS 650X wird vorausgesetzt, daß während des Einschaltens alle RESET-Eingänge des Prozessors und aller peripheren Schnittstellen-Bausteine auf L gehalten werden, bis die Versorgungsspannungen und Takte stabil anstehen. Dies Verfahren stellt sicher, daß die Peripherie-Anschlüsse in einem bekannten Zustand bleiben, bis im ganzen System die Anfangsbedingungen hergestellt sind und der Prozessor die Steuerung der Ausgangsleitungen übernehmen kann, d.h. das Systemprogramm laufen läßt.

Es muß erwähnt werden, daß in allen Bausteinen beim Einschalten der Stromversorgung der Inhalt der Speicher, Register usw. vollkommen zufällig ist. An den peripheren Ausgangsanschlüssen können zufällige Daten verhängnisvoll sein. Die einzige Methode, diese Ausgangsleitungen in einen bekannten Zustand zu zwingen, ist das Anlegen des Signals RESET. Der Entwickler kann dann sichergehen, daß der bekannte Zustand in den Peripherie-Geräten keine störenden Vorgänge verursacht. Die Wirkung des RESET auf die peripheren Bausteine wird bei der Untersuchung jedes einzelnen besprochen.

Das einzige Register im Prozessor, das in einen bekannten Zustand gebracht werden muß ist der Programmzähler. Dieses Register wählt die auszuführenden Befehle aus. Der RESET-Eingang veranlaßt den Programmzähler zum ersten Befehl im Systemprogramm zu gehen. Die Einzelheiten dieses Vorgangs sind in Abschnitt 1.4.1.2.11 besprochen.

Der RESET-Eingang hat auf die peripheren Schnittstellen-Bausteine eine sehr wichtige weitere Wirkung. Obwohl die Erkennung eines Interrupt-Signals für den Prozessor automatisch ist und nicht von der Software abhängt, wird die Reihenfolge der vom Prozessor ausgeführten Operationen zur vollständigen Bedienung eines Interrupts durch das Programm bestimmt. Bis die verschiedenen internen Register im Prozessor initialisiert sind, ist der Prozessor nicht bereit auf externe Interrupts richtig zu antworten. Der System-RESET muß daher alle externen Interrupt-Signale abschalten, bis sie vom Prozessor eingeschaltet werden. Dann kann der Programmierer sicherstellen, daß das System richtig auf seine Anfangsbedingungen gesetzt ist, bevor Interrupts eingeschaltet werden.

1.4 DIE MIKROPROZESSOREN

1.4.1 Der MCS 6502

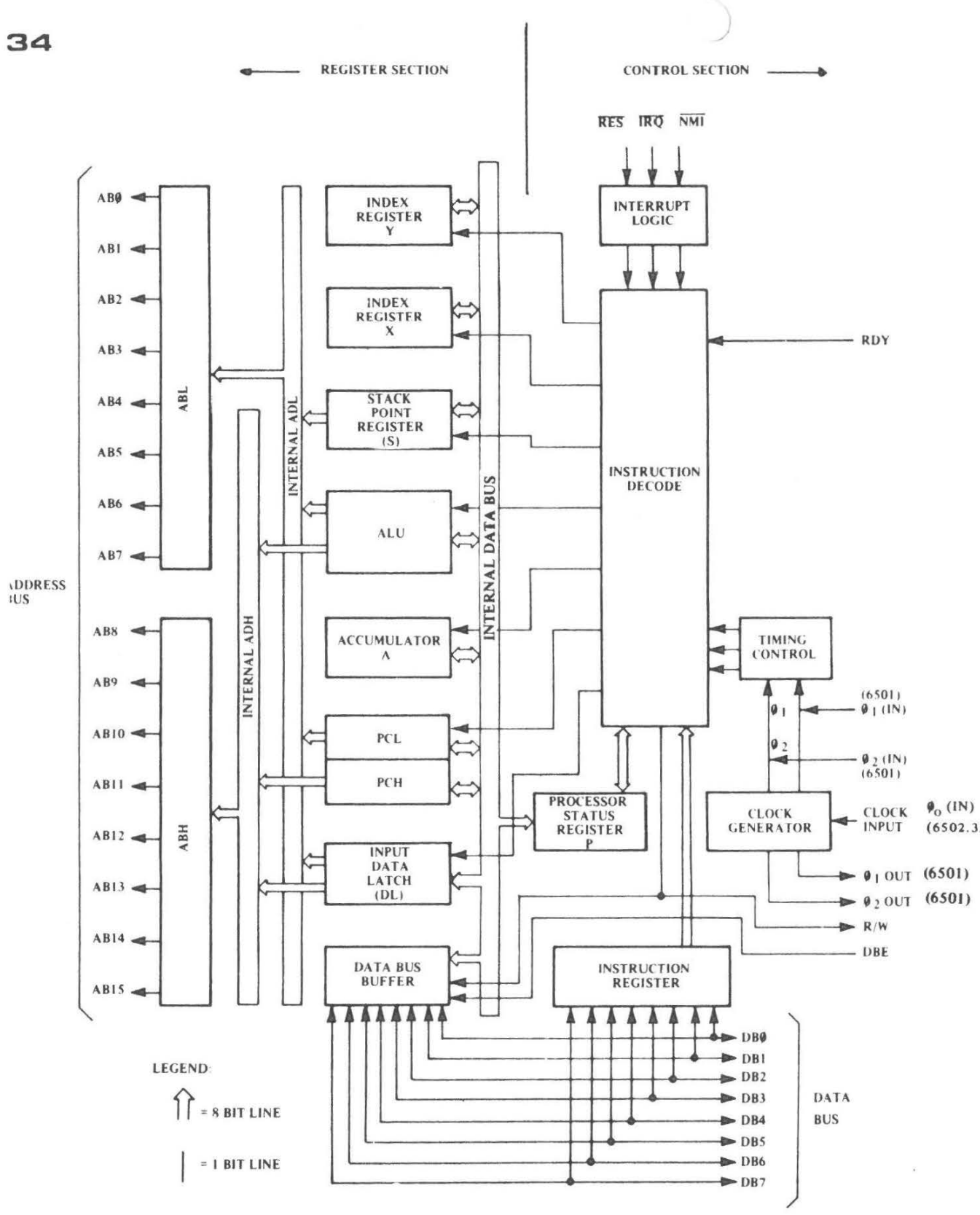
1.4.1.1 Einführung

Die internen Strukturen der Mitglieder der Mikroprozessor-Familie MCS 650X sind sehr ähnlich. Ein Blockschaltbild dieser Struktur zeigt Bild 1.11. Dieser Abschnitt beginnt mit einer Analyse dieses Blockschaltbildes, bespricht die Funktion der verschiedenen Register, Datenwege usw. Es folgt eine genaue Besprechung der Arbeitsweise der verschiedenen Anschlüsse.

Die interne Organisation des Prozessors gliedert sich in zwei Teile. Im allgemeinen werden die Befehle, die aus dem Programmspeicher stammen, durch eine Reihe von Datenübertragungen in einem Teil des Baustein, dem Registerteil, ausgeführt. Die Steuersignale für die Datenübertragung werden im anderen Teil, dem Steuerteil, erzeugt. Die Befehle kommen auf dem Datenbus in den Prozessor und werden im Befehls-Register gespeichert. Sie werden dann zusammen mit Zeitsignalen dekodiert, um die Steuersignale für die Register zu erzeugen.

Die Zeitsteuer-Einheit überwacht, welcher Zyklus gerade ausgeführt wird. In jedem Zyklus, in dem ein Befehl geholt wird, wird diese Einheit auf „ T 0 " gesetzt und beim Anfang jedes Takt 1 Impulses weitergeschaltet. Jeder Befehl fängt mit T 0 an und geht über T1, T2 usw., bis der gesamte Befehl ausgeführt ist. Jede Datenübertragung usw. im Registerteil wird durch die Kodierung des Inhalts von Befehls-Register und Zeitzähler ausgelöst.

Weitere Steuersignale für die Ausführung von Befehlen werden von der Interrupt-Logik und vom Prozessor Status-Register abgeleitet. Die Interrupt-Logik steuert die Schnittstelle des Prozessors zu den Interrupt-Eingängen, die der Prozessor erkennt und bedient.



NOTE: 1. CLOCK GENERATOR IS NOT INCLUDED ON MCS6501.
2. ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE MCS650X PRODUCTS.

MCS650X Internal Architecture

ABBILDUNG 1.11: MCS 650X INTERNER AUFBAU

Das Prozessor Status-Register enthält einen Satz von Flip-Flops, die zur Steuerung einiger Einzelheiten der Arbeitsweise dienen, die die Ergebnisse von arithmetischen und logischen Operationen im Prozessor anzeigen, und die den Status der entweder vom Prozessor erzeugten Daten oder zum Prozessor von außen übertragenen Daten anzeigen. Die eigentliche Arbeit des Prozessors wird im Registerteil ausgeführt. Daher wird dieser Teil jetzt genauer untersucht. Er besteht aus:

- Puffern für den Datenbus
- Register für Eingangsdaten (DL, Input Data Latch)
- Programmzähler (PCL, PCH, Program Counter)
- Akkumulator (A)
- Arithmetische Einheit (ALU, Arithmetic Logic Unit)
- Stapelzeiger (S, Stack Pointer)
- Index-Register (X, Y, Index Registers)
- Adress-Bus-Register (ABL, ABH, Adress Bus Latches)
- Prozessor-Status-Register (P)

Die Daten die vom Programmspeicher, dem Datenspeicher oder von Peripheriegeräten in den Prozessor kommen, erscheinen bei einer Arbeitsgeschwindigkeit von 1 MHz während der letzten 100 Nanosekunden des Taktes 2 auf dem Datenbus. Während dieser kurzen Zeit wird nicht versucht, tatsächlich mit diesen Daten zu arbeiten. Sie werden stattdessen einfach in das Eingangs-Daten-Register übertragen, um im nächsten Zyklus benutzt zu werden. Das Daten-Register hält die Daten auf dem Datenbus während jedes Taktes 2 fest. Sie können dann auf der internen Busse und von da auf eines der internen Register übertragen werden. Z.B. werden Daten die vom Speicher zum Akkumulator (A) übertragen werden auf den internen Datenbus gelegt und von da auf den Akkumulator übertragen. Wenn mit den Daten vom Speicher und dem Inhalt des Akkumulators eine arithmetische oder logische Operation ausgeführt wird, werden die Daten im Eingangs-Daten-Register wie vorher auf den internen Daten-Bus übertragen und von dort in die ALU.

Zur gleichen Zeit wird der Inhalt des Akkumulators auf einen Bus im Register-Teil und von da auf den zweiten Eingang der ALU übertragen. Die Ergebnisse der arithmetischen oder logischen Operation werden im nächsten Zyklus zum Akkumulator übertragen, zuerst auf den Bus und dann in den Akkumulator. Alle diese Datenübertragungen finden während des Taktes 2 statt.

Der Programmzähler (PCL, PCH,) liefert die Adresse, die den Prozessor durch aufeinanderfolgende Befehle im Programm führen. Jedes Mal wenn der Prozessor einen Befehl aus dem Programmspeicher holt wird der Inhalt von PCL auf die niederwertigen 8 Bit des Adress-Bus und der Inhalt von PBH auf die höherwertigen 8 Bit gelegt. Dieser Zähler wird jedes Mal, wenn ein Befehl oder Daten aus dem Programm-speicher geholt werden, um einen Schritt hochgezählt.

Der Akkumulator ist ein Mehrzweck-Register mit 8 Bit, das die Ergebnisse der meisten arithmetischen und logischen Operationen speichert. Außerdem enthält der Akkumulator eines der beiden Daten-worte, die in diesen Operationen verwendet werden.

Alle logischen und arithmetischen Operationen finden in der ALU statt, das schließt hoch- und herunterzählen der internen Register ein (außer PCL und PCH). Die ALU kann jedoch Daten nicht für mehr als einen Zyklus speichern. Wenn am Anfang eines Zyklus Daten an den Eingang der ALU gelegt werden, wird das Ergebnis im nächsten Zyklus immer in eines der Speicher-Register oder im externen Speicher geladen. Jedes Bit der ALU hat zwei Eingänge. Diese Ein-gänge können mit verschiedenen internen Bussen oder mit der logisch 0 verbunden werden. Die ALU erzeugt dann die Funktionen SUM, AND, OR usw. aus den Daten der beiden Eingänge.

Der Stapelzeiger (S) und die zwei Index-Register (X und Y) bestehen aus je acht einfachen Flip-Flops. Diese Register speichern die Daten zur Errechnung der Adressen im Datenspeicher. Ihre genaue Arbeitsweise wird in der Programmier-Fibel im einzelnen besprochen.

Die Puffer für den Adress-Bus (ABL, ABH) bestehen aus einem Satz von Flip-Flops und TTL-kompatiblen Treibern. Sie speichern die Adressen, die zum Zugriff zu den peripheren Bausteinen (ROM, RAM, und Eingangs/Ausgang) dienen.

1.4.1.2 Anschlußbelegung des MCS 6502

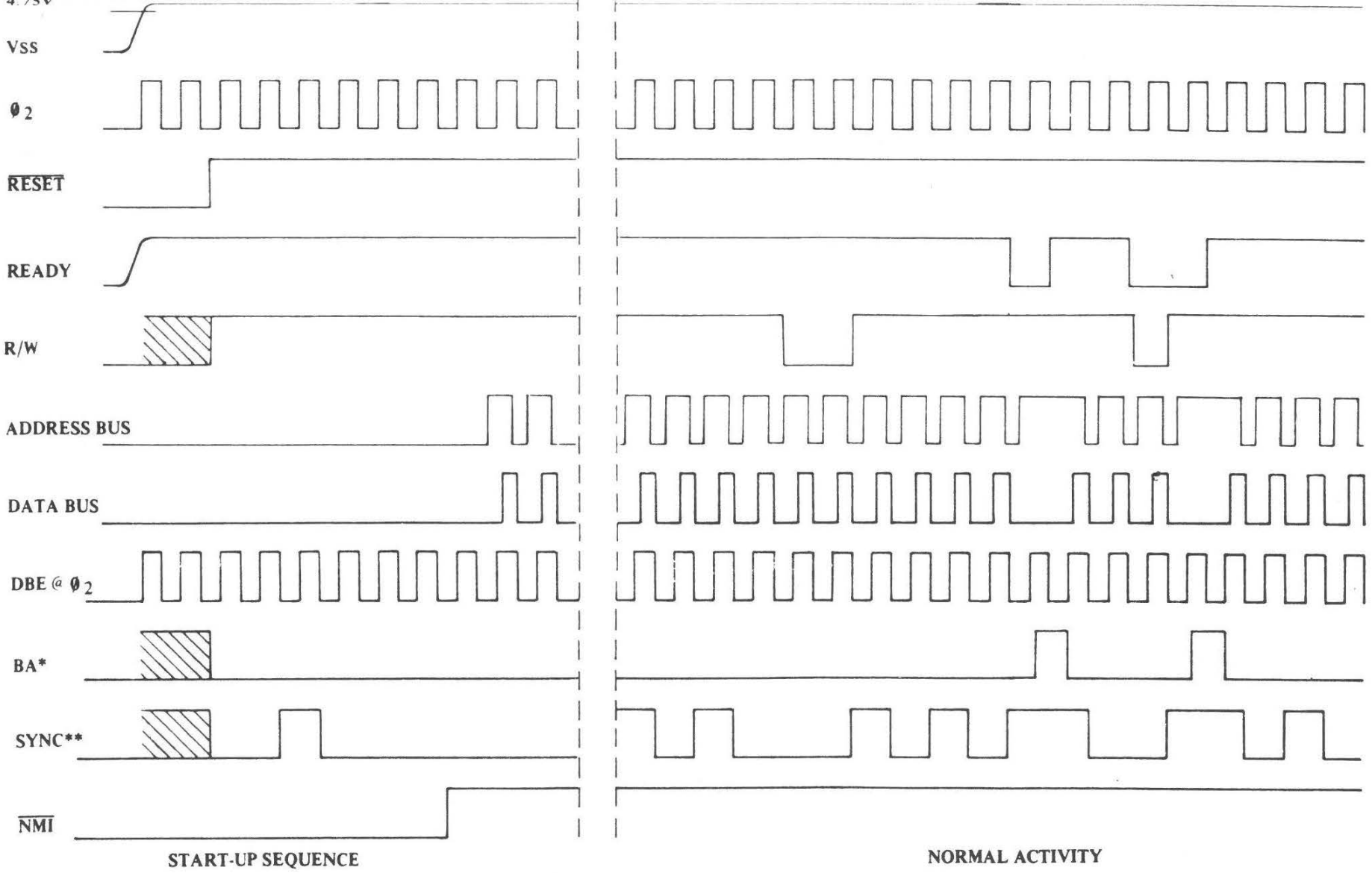
Bild 1.15 zeigt die Anschlußbelegung des MCS 6502. Diese Anschlüsse und ihre Benutzung im Mikrocomputer-System sind unten einzeln besprochen.

1.4.1.2.1 Vcc, Vss - Stromversorgungs-Leitungen

Die Anschlüsse Vcc und Vss sind die einzigen Stromversorgungs-Anschlüsse zum Baustein. Die Versorgungsspannung an Stift 8 ist + 5,0 V DC $\pm$ 5 %. Der absolute Grenzwert am Vcc-Eingang ist + 7,0 V DC.

1.4.1.2.2 AB00 - AB15 - Adressbus

Die Adressbus-Puffer der MCS 650X Familie sind Gegentakt-Treiber. Sie können eine Last von mindestens 130 pf und 1 Standard-TTL-Last treiben. Der Adressbus enthält immer bekannte Daten, die in Anhang A ausgeführt. Zur Adressierung wird eine Adresse auf den Adressbus gelegt, der entweder in einer Programmfolge auf derselben Page im Programmspeicher oder an einem bekannten Punkt im RAM ist. Die genaue Arbeitsweise dieses Bus kann der Entwickler dem Anhang A entnehmen.



*BA IS AVAILABLE ON MCS6501 ONLY
**SYNC IS AVAILABLE ON MCS6502 ONLY

MCS650X System Timing Diagram

ABBILDUNG 1.13: MCS 650X ZEITDIAGRAMM DES SYSTEMS

Die verschiedenen Prozessoren unterscheiden sich etwas in der Zahl der Adressleitungen. Der MCS 6504 hat 13 Adressleitungen (AB00 - AB12) und der MCS 6503 und MCS 6505 haben je 12 (AB00 - AB11). Der MCS 6504 kann also 8.192 Byte Speicher adressieren und der MCS 6503 und MCS 6505 4.096 Byte. Für die kleinen, kostenempfindlichen Systeme, in denen diese Bausteine ihre Hauptanwendung finden, sollte dieser Adressraum mehr als ausreichend sein.

Der Zeitablauf auf dem Adressbus ist für alle Prozessoren genau der gleiche. Die Adresse ist für 300 ns in dem Takt 1 Impuls gültig und bleibt bis zum nächsten Takt 1 Impuls stabil (für 1 MHz Takt). Diese Angaben ändern sich nur für Prozessoren, die zum Betrieb mit höheren Taktfrequenzen vorgesehen sind. Bild 1.13 zeigt die Beziehung des Adress-Bus zu anderen typischen Signalen. Es ist wegen der kleineren Zahl von Adress-Leitungen bei den Prozessoren mit 28 Anschlüssen möglich, ein Programm zu schreiben, das den Zugriff zu nicht vorhandenem Speicher-Adressraum versucht, d.h. die Adress-Bits 13, 14 oder 15 liegen auf H. Diese oberen Adress Bits im Programm werden vernachlässigt und das Programm fällt in den bestehenden Adressraum zurück. Die Speicherbenutzung muß deswegen gut geplant werden wenn Bausteine mit großer Adressierfähigkeit verwendet werden, damit die Adressiergrenzen eines Bausteins mit kleinerer Adressierfähigkeit eingehalten werden.

1.4.1.2.3 DB0 - DB7 - Daten-Bus

Der Prozessor-Datenbus ist für alle derzeit lieferbaren Prozessoren und für die Software-kompatiblen Prozessoren, die in der nahen Zukunft herausgebracht werden, genau der gleiche. Alle Befehle und Datenübertragungen zwischen dem Prozessor und dem Speicher finden auf diesen Leitungen statt. Die Puffer für die Daten-Bus Leitungen haben drei Ausgangszustände (three-state) das ist wegen des Betriebes in zwei Richtungen nötig.

Jeder Anschluß des Daten-Bus ist mit einem Eingangs- und einem Ausgangspuffer verbunden. Der Ausgangspuffer verbleibt im hochohmigen Zustand, außer wenn der Prozessor Daten in einem oder aus einem der Unterstützungs-Bausteinen überträgt. Alle Datenübertragungen zwischen den Bausteinen finden während des Taktes 2 statt. Während des Taktes 1 „ schwebt " der gesamte Daten-Bus, d.h. alle Treiberausgänge sind hochohmig.

Der Datenbus-Puffer ist ein Gegentakt-Treiber, der 130 pf und eine Standard TTL-Last bei der Arbeitsgeschwindigkeit treibt. Bei einer Taktfrequenz von 1 MHz müssen die Daten 100 ns vor dem Ende des Taktes 2 stabil sein. Das gilt für Übertragungen in beide Richtungen. Bild 1.13 zeigt das Verhältnis von Datenbus zu den anderen Signalen.

1.4.1.2.4 R/W - Schreiben/Lesen (Read/Write)

Mit dem Anschluß R/W steuert der Prozessor die Richtung der Datenübertragung zwischen sich und den Unterstützungs-Bausteinen. Dieser Anschluß liegt auf H, außer wenn der Prozessor in den Speicher oder in einem peripheren Schnittstellen-Baustein einschreibt.

Alle Übergänge an diesem Anschluß geschehen während des Taktes 1 (gleichzeitig mit den Adress-Leitungen). Dadurch ist eine vollkommene Steuerung des Datenüberganges möglich, der während des Taktes 2 stattfindet.

Der R/W - Puffer ist den Adress-Puffern ähnlich.Er kann 130 pf und eine Standard TTL-Last bei Arbeitsgeschwindigkeit treiben. Bild 1.13 zeigt wieder die Zeitverhältnisse auf der Leitung R/W.

1.4.1.2.5 Das SYNC-Signal

Das SYNY-Signal zeigt die Zyklen an, in denen der Prozessor einen OP-CODE holt. Die Leitung SYNC geht während des Taktes 1 des Holens eines OP-CODE auf H und bleibt für den Rest des Zyklus auf H. Wenn die Leitung RDY während des Taktes 1 auf L gezogen wird, in der die Leitung SYNC auf H gegangen ist, hält der Prozessor in seinem derzeitigen Zustand an. Er bleibt in diesem Zustand bis die Leitung RDY auf H geht. Das Signal SYNC kann auf diese Weise zur Steuerung von RDY benutzt werden, um die Ausführung einzelner Befehle zu erzielen. Diese Anwendung wird in Kapitel 2 besprochen. Bild 1.18 enthält das Zeitdiagramm für dieses Signal.

1.4.1.2.6 RDY - Fertig (Ready)

Der Eingang RDY verzögert die Ausführung jedes beliebigen Zyklus, währenddessen die Leitung RDY auf L gezogen wird. Diese Leitung sollte während des Taktes 1 wechseln. Der Wechsel wird dann während des nächsten Taktes 2 erkannt und schaltet die Ausführung des gerade laufenden internen Zyklus ein oder aus. Diese Ausführung geschieht normalerweise während des nächsten Taktes 2. Das Zeitdiagramm zeigt Bild 1.13.

Der Hauptzweck der Leitung RDY ist die Verzögerung der Ausführung eines Programm-Hol-Zyklus, bis Daten vom Speicher verfügbar sind. Das findet direkte Anwendung in Prototyp-Systemen, die UV-löschbare PROM oder EAROM (elektrisch löschbar) verwenden. Beide Speichertypen haben relativ große Zugriffszeiten und müssen mit der Funktion RDY betrieben werden, wenn der Prozessor mit voller Geschwindigkeit arbeiten soll. Ohne die Funktion RDY wäre eine Herabsetzung der Taktfrequenz des Systems nötig.

Die Funktion RDY hält den Prozessor nicht an in einem Zyklus in dem ein Schreib-Vorgang ausgeführt wird. Wenn die Leitung RDY von H nach L geht, während der Prozessor in einem Schreib-Zyklus ist, führt er diesen zu Ende aus und hält dann im nächsten Lese-Zyklus (R/W = 1) an.

1.4.1.2.7 $\overline{\text{NMI}}$ - Nicht-maskierbarer Interrupt (Non-Maskable Interrupt)

Der Eingang $\overline{\text{NMI}}$ unterbricht den Prozessor immer nachdem er den gerade laufenden Befehl ausgeführt hat. Dieser Interrupt ist nicht „ maskierbar ", d.h. der Prozessor kann seine Erkennung auf keine Weise verhindern.

Der Eingang $\overline{\text{NMI}}$ spricht auf einen Übergang von H zu L an. Er kann dann ohne Einfluß auf die Arbeitsweise des Prozessors und ohne einen weiteren Interrupt unbegrenzte Zeit auf L bleiben. Der Prozessor erkennt einen weiteren Interrupt erst beim nächsten Übergang von H zu L. Zur Erkennung muß das Signal $\overline{\text{NMI}}$ für mindestens zwei Taktperioden auf L liegen. Daraufhin werden neue Vektoren zur Programmzählung geholt.

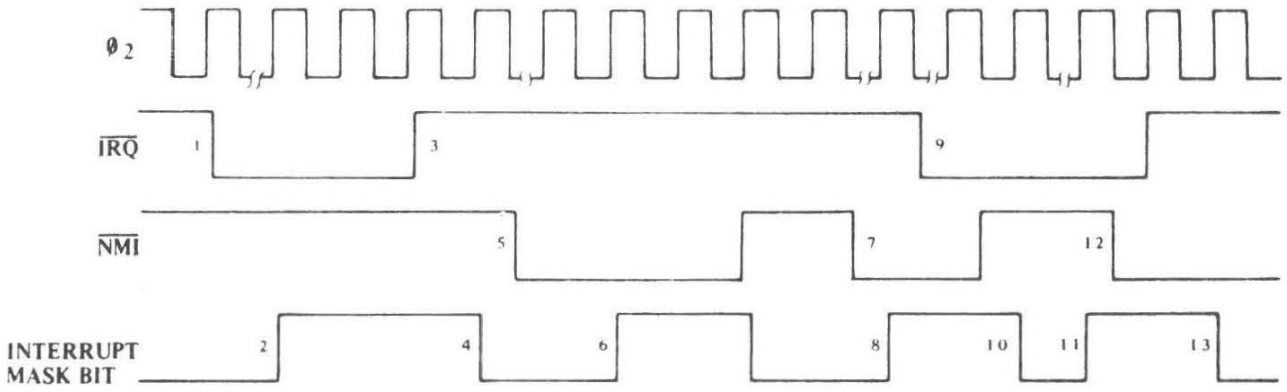
1.4.1.2.8 $\overline{\text{IRQ}}$ - Interrupt-Anforderung (Interrupt Request)

Der Eingang $\overline{\text{IRQ}}$ reagiert in sehr ähnlicher Weise wie $\overline{\text{NMI}}$. Diese Funktion kann jedoch durch ein Bit im Prozessor-Status-Register ein- und ausgeschaltet werden. So lange das I-Flip-Flop (interrupt inhibit flag) auf H steht, beeinflußt das Signal am Anschluß $\overline{\text{IRQ}}$ den Prozessor nicht.

Der Anschluß $\overline{\text{IRQ}}$ ist nicht flankengesteuert. Der Prozessor wird so lange unterbrochen, wie das I-Flip-Flop auf L steht und das Signal am Eingang $\overline{\text{IRQ}}$ auf L liegt. Deswegen muß das Signal $\overline{\text{IRQ}}$ auf L gehalten werden, bis es erkannt worden ist. D.h. der Prozessor führt den laufenden Befehl zu Ende aus.

Wenn I gesetzt ist, während $\overline{IRQ}$ auf L liegt, wird der Interrupt nicht erkannt, bis I durch Software-Steuerung rückgesetzt ist. Damit der Prozessor einen Interrupt nicht mehr als einmal erkennt, wird I automatisch während des letzten Zyklus gesetzt, bevor der Prozessor mit der Ausführung der Interrupt-Software anfängt.

Als letzte Bedingung muß der Interrupt-Eingang gelöscht sein, bevor das I-Flip-Flop rückgesetzt wird. Wenn mehr als ein aktiver Interrupt diese beiden Leitungen betreibt (logisches ODER) wird empfohlen beide Interrupts zu bedienen und zu löschen, bevor das I-Flip-Flop gelöscht wird. Wenn die Interrupts einzeln gelöscht werden und das I-Flip-Flop nach jedem zurückgesetzt wird, erkennt der Prozessor einfach jeden noch aktiven Interrupt und wird ihn richtig verarbeiten. Das geschieht jedoch langsamer wegen der zur Rückkehr von einem Interrupt benötigten Zeit, bevor der andere erkannt wird. Bei Einhaltung des oben empfohlenen Verfahren wird jeder Interrupt nur einmal erkannt und bearbeitet. Bild 1.14 zeigt mehrere Beispiele von Interrupts zur Erkennung des Interrupt ($\overline{IRQ}$ und $\overline{NMI}$) durch den Prozessor und die Auswahl der Interrupts durch den Prozessor während sich überlappender Anforderungen.



Examples of Interrupt Recognition by MCS650X

ABBILDUNG 1.14: BEISPIELE DER INTERRUPT-ERKENNUNG BEIM MCS 650X

Jeder größere Vorgang, der den Mikroprozessor beeinflußt ist in Bild und den entsprechenden Erklärungen unten gleich beziffert.

Nummer des Ereignisses	Vorgang im System
1.	Der Prozessor führt das Hauptprogramm aus und $\overline{IRQ}$ geht auf L.
2.	Nach Ausführung des laufenden Befehl erkennt der Prozessor den Interrupt, speichert den Inhalt von PC und P im Stapel und setzt I, während er den Interrupt-Sektor holt.
3.	Nach der Bedienung des Interrupt muß $\overline{IRQ}$ zurückgesetzt werden, bevor das Maskierungs-Bit für den Interrupt zurückgesetzt wird, um einen doppelten Interrupt zu vermeiden.
4.	Bevor der Prozessor die normale Ausführung des Programms wieder aufnimmt, wird das Maskierungs-Bit für den Interrupt wieder auf L gesetzt.
5.	$\overline{NMI}$ geht jetzt auf L und zeigt damit einen nichtmaskierbaren Interrupt Anforderung an.
6.	Der Interrupt $\overline{NMI}$ wird erkannt und auf die gleiche Weise wie $\overline{IRQ}$ bedient.
7.	Der Prozessor hat seine normale Arbeitsweise wieder aufgenommen. Dann geht $\overline{NMI}$ wieder auf L und fordert damit einen Interrupt an.
8.	Als Antwort auf die Anforderung $\overline{NMI}$ wird das Maskierungs-Bit für den Interrupt auf H gesetzt.

9. Jetzt ist $\overline{IRQ}$ auf L gegangen, um eine Interrupt-Anforderung anzuzeigen. Diese Anforderung wird übersehen, da der Interrupt $\overline{NMI}$ bedient wird und die Interrupt-Maske gesetzt ist.

10. Das Maskierungs-Bit für den Interrupt wird nach Bedienung des Interrupt $\overline{NMI}$ zurückgesetzt.

11. Der Prozessor kann jetzt das Signal $\overline{IRQ}$, das noch auf L liegt, erkennen und tut dies durch Setzen des Maskierungs-Bit für den Interrupt.Während der Bedienung von $\overline{IPQ}$ geht $\overline{NMI}$ von H auf L. Der Prozessor vollendet dann den laufenden Befehl und übersieht den Interrupt $\overline{IRQ}$, um $\overline{NMI}$ zu bedienen. $\overline{NMI}$ wird ohne Rücksicht auf den Zustand des Maskierungs-Bit für den Interrupt bedient.

13. Nach Beendigung der Routine für den $\overline{NMI}$-Interrupt nimmt der Prozessor die Ausführung der $\overline{IRQ}$-Routine wieder auf, selbst wenn $\overline{IRQ}$ in der zwischenzeit wieder auf H liegt.

1.4.1.2.9 $\overline{RES}$ - Rücksetzen (RESET)

Das Signal $\overline{RES}$ setzt den Mikroprozessor vom stromlosen Zustand in die Anfangsbedingungen (Initialisieren). Während des Hochlaufens der Stromversorgung wird diese Leitung auf L gehalten und Schreiben vom Prozessor wird verhindert. Wenn die Leitung auf H geht, verzögert der Prozessor um 6 Zyklen und holt den neuen Programm-Zählvektor von bestimmten Stellen im Speicher (PCL von der Stelle FFFC und PCH von der Stelle FFFD). Das ist der Anfang des Benutzerprogramms. Man muß davon ausgehen, daß die inneren Zustände der Maschine unbekannt sind, wenn die Rücksetzleitung einmal auf L gezogen worden ist und wieder auf H geht.

Dann müssen alle Register während einer Folge zum Wiederstart neu initialisiert werden. Bild 1.13. zeigt das Zeitdiagramm für den Rücksetz-Ablauf.

1.4.1.2.10 Takt-Erzeugung -

Die Prozessoren können mit einem extern erzeugten Takt betrieben werden, entweder einem Einphasen-Takt mit TTL-Pegel, einem Quarz-Oszillator oder einem RC-Netzwerk.

Die Bilder 1.16 und 1.17 zeigen die Anordnung zur Bestimmung der Schwingfrequenz mit einem Quarz oder einem RC-Netzwerk.

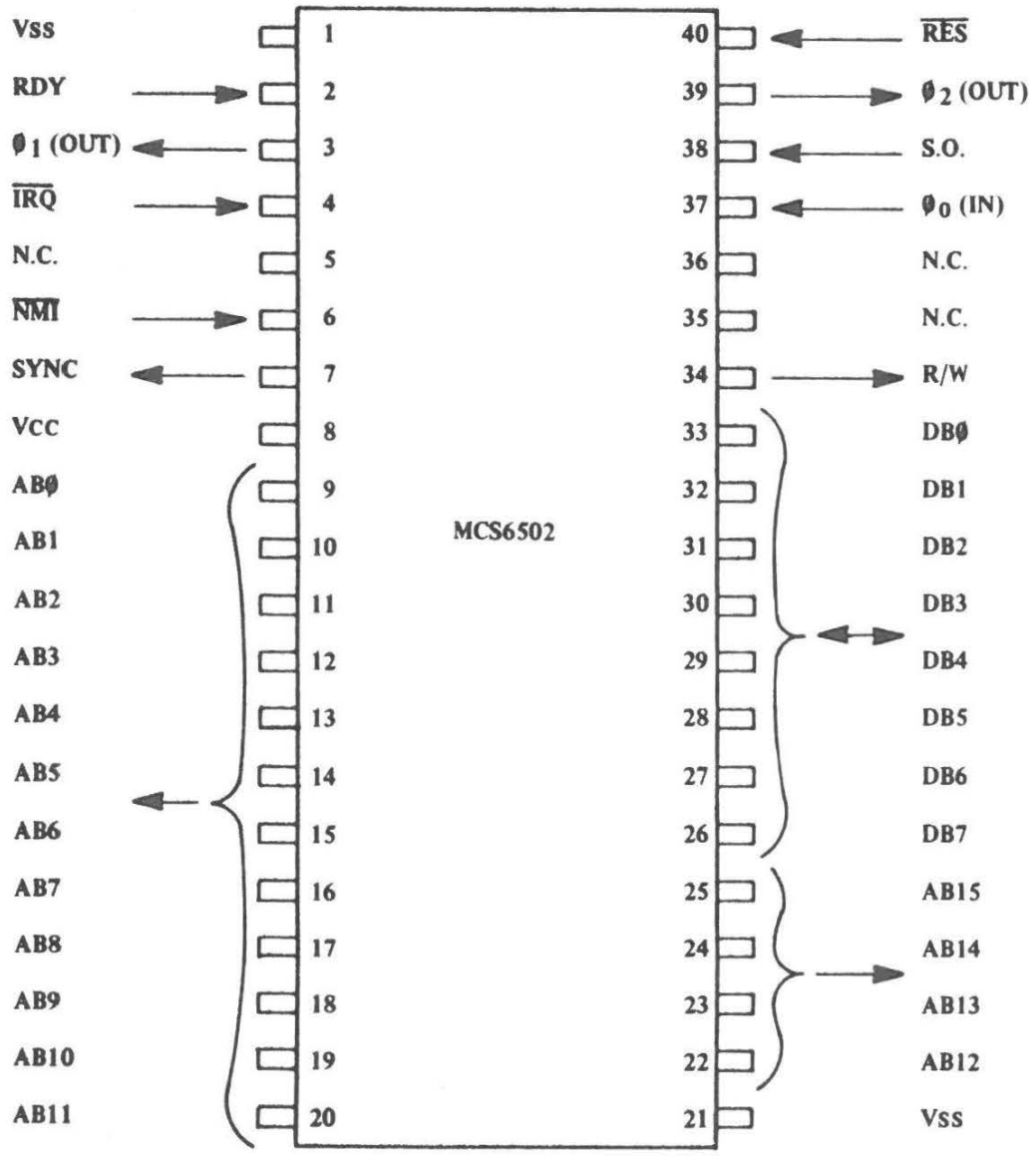
Bild 1.16 zeigt den Betrieb mit Quarz, in dem die Schwingfrequenz durch den Quarz zusammen mit dem RC-Netzwerk festgelegt wird. Bild 1.17 zeigt diesselbe Verbidnung, nur der Quarz ist aus der Schaltung entfernt. Die Werte für den Rückkoppelungs-Wiederstand R_F und den Rückkoppelungs-Kondensator C_F sind in beiden Betriebsarten verschieden. Genaue Daten für die Werte von R_F und C_F sind in den Datenblättern für den MCS 6502 angegeben. Der Taktgenerator arbeitet mit Kombinationen von R_F im Bereich von 0 bis 500 Kilo-Ohm und C_F von 2 bis 12 Piko-Farad. Für eine genaue Beschreibung wird der Leser auf das Datenblatt für den MCS 6502 verwiesen.

Das in Abschnitt 3.1.1 beschriebene Verfahren auf dem MCS 6500-Bus ist unabhängig vom Ort des Oszellators anwendbar. Die Zeiten der Takte, die die interne Arbeitsweise des Prozessors steuern, müssen dem Takt 2 aus Anschluß 39 sehr nahe kommen. Für die richtige Ausführung von Datenübertragungen dürfen sie um nicht mehr als zwei TTL-Verzögerungszeiten für die Pufferung des Taktes abweichen. In Systemen, in denen die Taktgeneratoren mit einer TTL-Impulsfolge getrieben werden, ist es wichtig daß diese Eingangs-Impulsfolge nur dann zur Steuerung der Peripherie-Bausteine benutzt wird, wenn der richtige Zeitablaufs des Taktes 2 sichergestellt ist.

1.4.1.2.11 S.O. - Setzen des Überlaufs (Set Overflow)

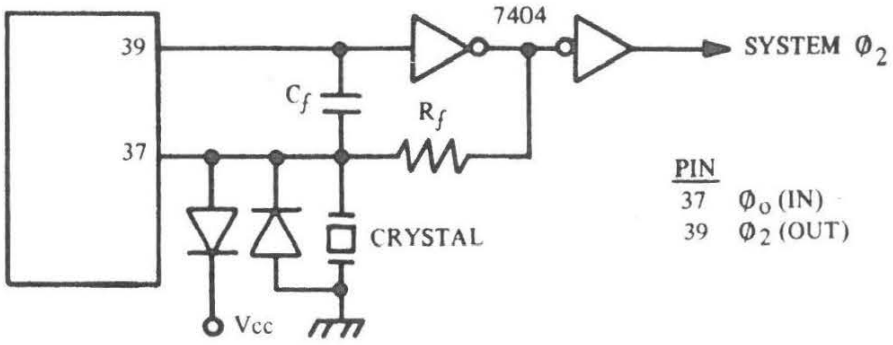
Dieser Stift setzt das Überlauf-Flip-Flop bei einer negativen TTL-Flanke (Übergang von H zu L). Das ist zur Zusammenarbeit mit einem zukünftigen Eingangs/Ausgangs-Baustein vorgesehen und sollte normalerweise nicht benutzt werden. Bei Verwendung muß beachtet werden daß die arithmetischen Operationen das Überlauf-Flip-Flop auch beeinflussen.

Bild 1.20 zeigt die Stiftbelegung für die drei Prozessoren. Wegen der beschränkten Anschlußzahl bestehen Einschränkungen, die zu Kompromissen zwischen Steuersignalen und Adressiermöglichkeit führen. Die Prozessoren mit 28 Stiften haben wie der MCS 6502 Oszillatoren und Takttreiber auf dem Baustein. Die Takterzeugung mit Quarz wird in Bild 1.21 und die mit RC-Netzwerk in Bild 1.22 gezeigt. Die genauen Werte von Rückkopplungs-Wiederstand und Kondensator sind im entsprechenden Datenblatt angegeben.



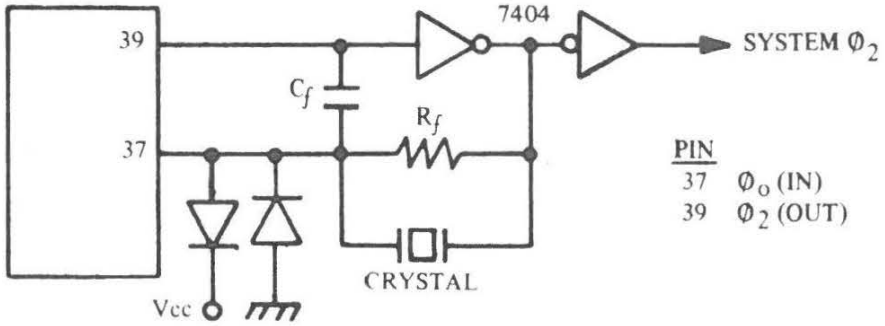
MCS6502 Pinout Designation

ABBILDUNG 1.15: MCS 6502 BESCHREIBUNG DER STIFTBELEGUNG



MCS6502 Parallel Mode Crystal Controlled Oscillator

ABBILDUNG 1.16A: MCS 6502 TAKTGENERATOR MIT QUARZ IN SERIENRESONANZ

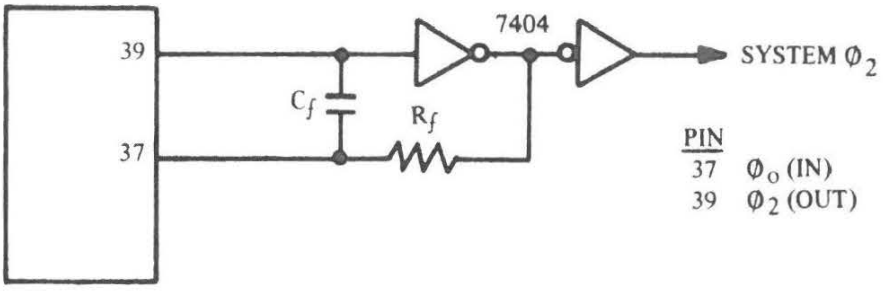


MCS6502 Series Mode Crystal Controlled Oscillator

ABBILDUNG 1.16B: MCS 6502 TAKTGENERATOR MIT QUARZ IN PARALLELRESONANZ

MCS6502 Time Base Generation - Crystal Controlled

ABBILDUNG 1.16: MCS 6502 TAKTGENERATOR, QUARZ IN PARALLELRESONANZ



MCS6502 Time Base Generator – RC Network

ABBILDUNG 1.17: MCS 6502 TAKTGENERATOR MIT RC-GLIED

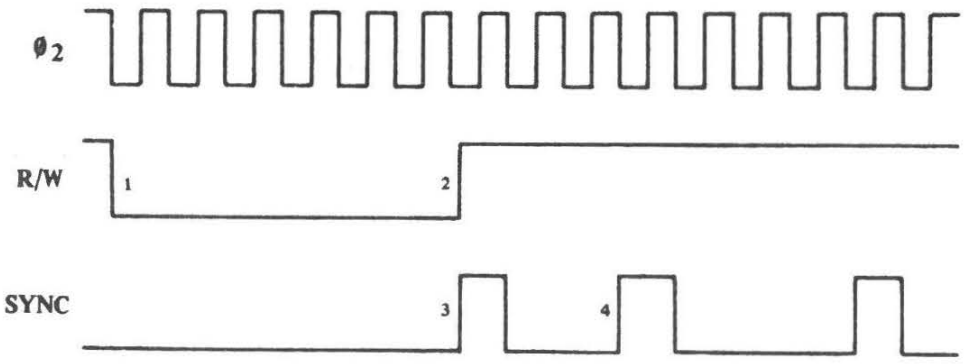


ABBILDUNG 1.18: MCS 6502 SYNC-SIGNAL

1. Während eines Schreibzyklus des Mikroprozessors liegt das Signal R/W auf L und der SYNC-Impuls tritt nicht auf.

2. Das Signal R/W geht auf H und zeigt damit den Anfang eines Lesezyklus des Prozessors an.

3. Zu Anfang des Lesezyklus wird ein SYNC-Impuls erzeugt. Dieser Impuls ist einen Zyklus lang. Er zeigt an, daß der Prozessor einen OP CODE aus dem Speicherfeld liest. In diesem Fall liegt der SYNC-Impuls für einen Zyklus auf H, während der Prozessor den OP CODE liest.

4. Der Prozessor gibt einen weiteren SYNC-Impuls aus und zeigt damit an, daß der vorhergehende Befehl ausgeführt ist und ein neuer OP CODE geholt wird. In diesem Fall sind drei weitere Zyklen nötig um diesen Befehl zu beenden, bevor der nächste SYNC-Impuls erzeugt wird. Der SYNC-Impuls ist aperiodisch. D.h. er ist eine Funktion des Programms und der sich daraus ergebenden Längen der Befehle und der Adressierarten.

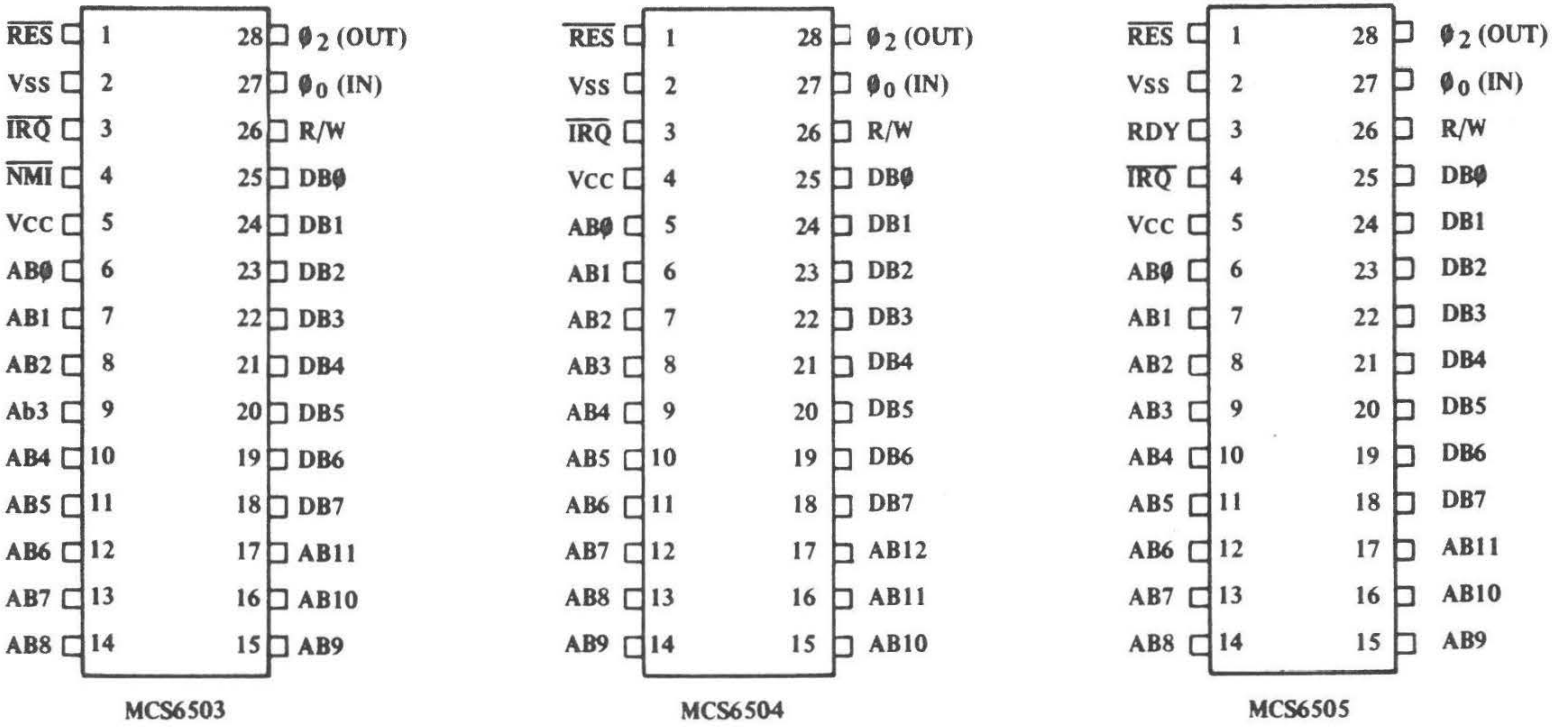
MCS 6502 SYNC-Signal
Bild 1.18

The operation of each function is exactly the same as on the MCS6502.

Features	MCS6503	MCS6504	MCS6505
Addressing Capability	4096 Bytes (AB00 - AB11)	8192 Bytes (AB00 - AB12)	4096 Bytes (AB00 - AB11)
Interrupt Request Capability	$\overline{\text{IRQ}}$, $\overline{\text{NMI}}$	$\overline{\text{IRQ}}$	$\overline{\text{IRQ}}$
"Ready" Signal	--	--	RDY
Timing Signals Required	Single Phase TTL Level $\emptyset_0$(IN), or Crystal or RC	Single Phase TTL Level $\emptyset_0$(IN), or Crystal or RC	Single Phase TTL Level $\emptyset_0$(IN), or Crystal or RC
Other Control Signals	$\overline{\text{RES}}$, R/W	$\overline{\text{RES}}$, R/W	$\overline{\text{RES}}$, R/W

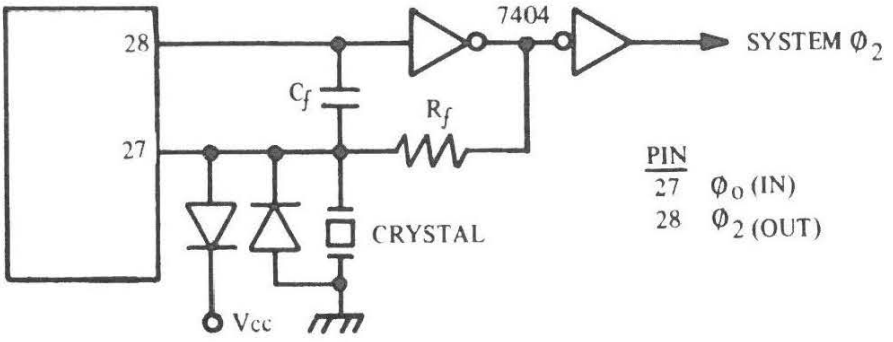
Functional Features of MCS6503, MCS6504, MCS6505

ABBILDUNG 1.19: MCS 6503, 6504, 6505 FUNKTIONSBESCHREIBUNG

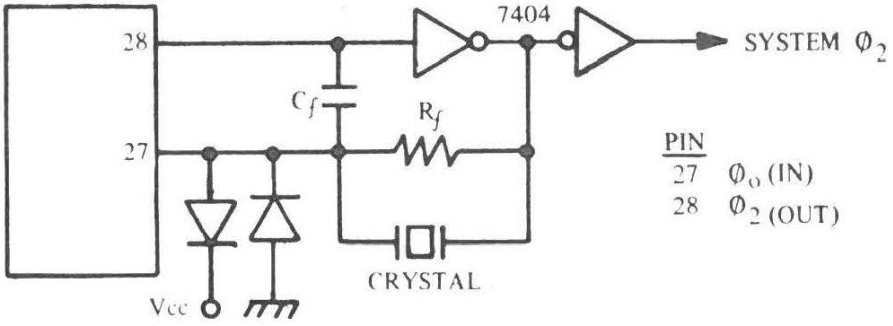


MCS6503, MCS6504, MCS6505 Pinout Designations

ABBILDUNG 1.20: MCS 6503, 6504, 6505 STIFTBELEGUNG



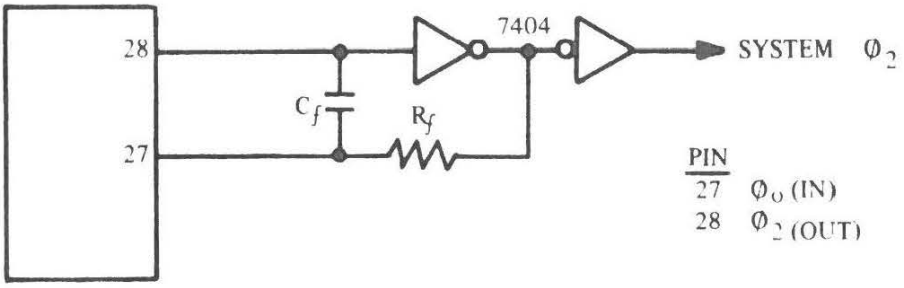
MCS6503,4,5 Parallel Mode Crystal Controlled Oscillator



MCS6503,4,5 Series Mode Crystal Controlled Oscillator

MCS6503, MCS6504, MCS6505 Time Base Generation Crystal Controlled

ABBILDUNG 1.21: MCS 6503, 6504, 6505 TAKTGENERATOR MIT QUARZ IN SERIENRESONANZ



MCS6503, MCS6504, MCS6505 Time Base Generation RC Network

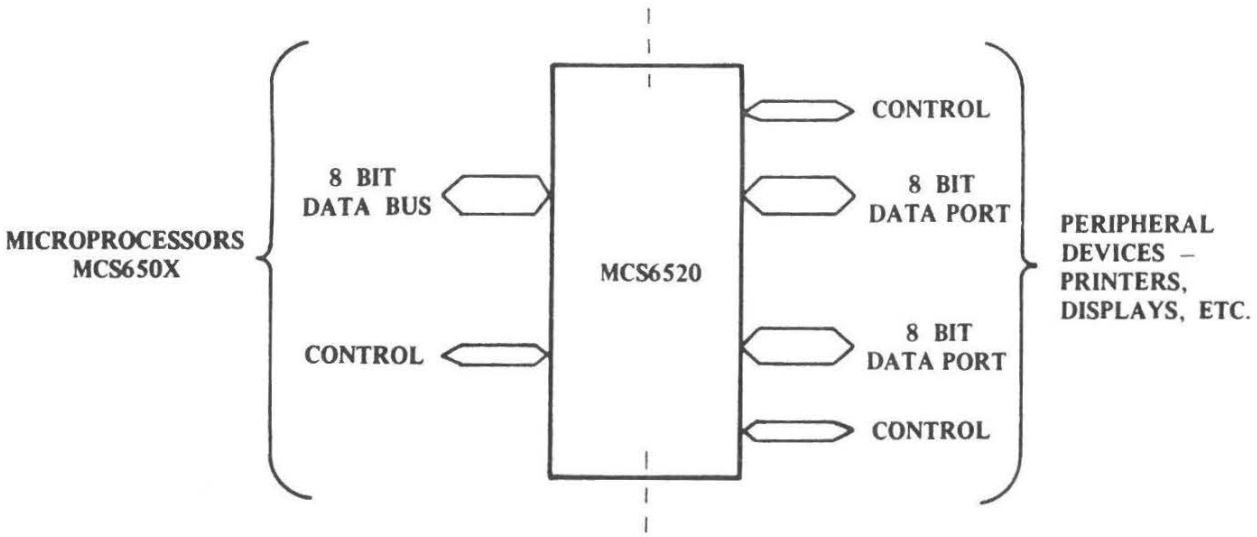
ABBILDUNG 1.22: MCS 6503, 6504, 6505 TAKTGENERATOR MIT QUARZ IN PARALLELRESONANZ

1.5 PERIPHERAL INTERFACE DEVICE – MCS6520

1.5.1 Introduction

The MCS6520 is a direct pin for pin replacement for the Motorola MC6820 Peripheral Interface Adapter, the "PIA". As such, it meets all of the "PIA" electrical specifications and is totally hardware compatible with the MC6820.

The MCS6520 is an I/O device which acts as an interface between the microprocessor and peripherals such as printers, displays, keyboards, etc. The prime function of the MCS6520 is to respond to stimulus from each of the two worlds it is serving. On the one side, the MCS6520 is interfacing with peripherals via two eight-bit bi-directional peripheral data ports. On the other side, the device interfaces with the microprocessor through an eight-bit data bus; this is the same data bus discussed at length in Section 1.3.1. It is, therefore, simplest to view the basic function of the MCS6520 as in the block diagram of Figure 1.23.



Basic MCS6520 Interface Diagram

ABBILDUNG 1.23: MCS 6520 SCHALTBILD DER SCHNITTSTELLE

1.5 BAUSTEIN FÜR PERIPHERE SCHNITTSTELLE - MCS 6520

1.5.1 Einführung

Der MCS 6520 ist ein direkter Ersatz für den Motorola MC 6820 Periphal Interface Adapter dem „PIA". Im Anschlußbild und elektrischen Daten stimmt er mit dem „PIA" überein und ist mit dem MC 6820 vollkommen hardwarekompatibel.

Der MCS 6520 ist ein Eingangs- Ausgangs-Baustein, der als Schnittstelle zwischen dem Mikroprozessor und Peripheriegeräten wie Druckern, Anzeigen, Tastaturen usw. dient. Der Hauptzweck des MCS 6520 ist es, auf Reize aus beiden Welten, die es bedient, zu reagieren. Auf der einen Seite stellt der MCS 6520 die Verbindung zu Peripheriegeräten über zwei 8-Bit periphere Datenkanäle für zwei Richtungen her. Auf der anderen Seite stellt der Baustein die Verbindung mit dem Mikroprozessor über einen 8-Bit Datenbus her. Dies ist der Datenbus, der in Abschnitt 1.3.1. genau besprochen worden ist. Es ist daher am einfachsten, die grundlegende Arbeitsweise des MCS 6520, die im Blockschaltbild Abb. 1.23 zu betrachten. Zusätzlich zu den oben beschriebenen Leitungen bietet der MCS 6520 vier Interrupt Eingangs/periphere Steuerleitungen und die für einfache, wirkungsvolle Steuerung peripherer Interrupts nötige Logik. Zur Verbindung des MCS 650X Mikroprozessors mit den meisten peripheren Geräten ist keine weitere Logik nötig.

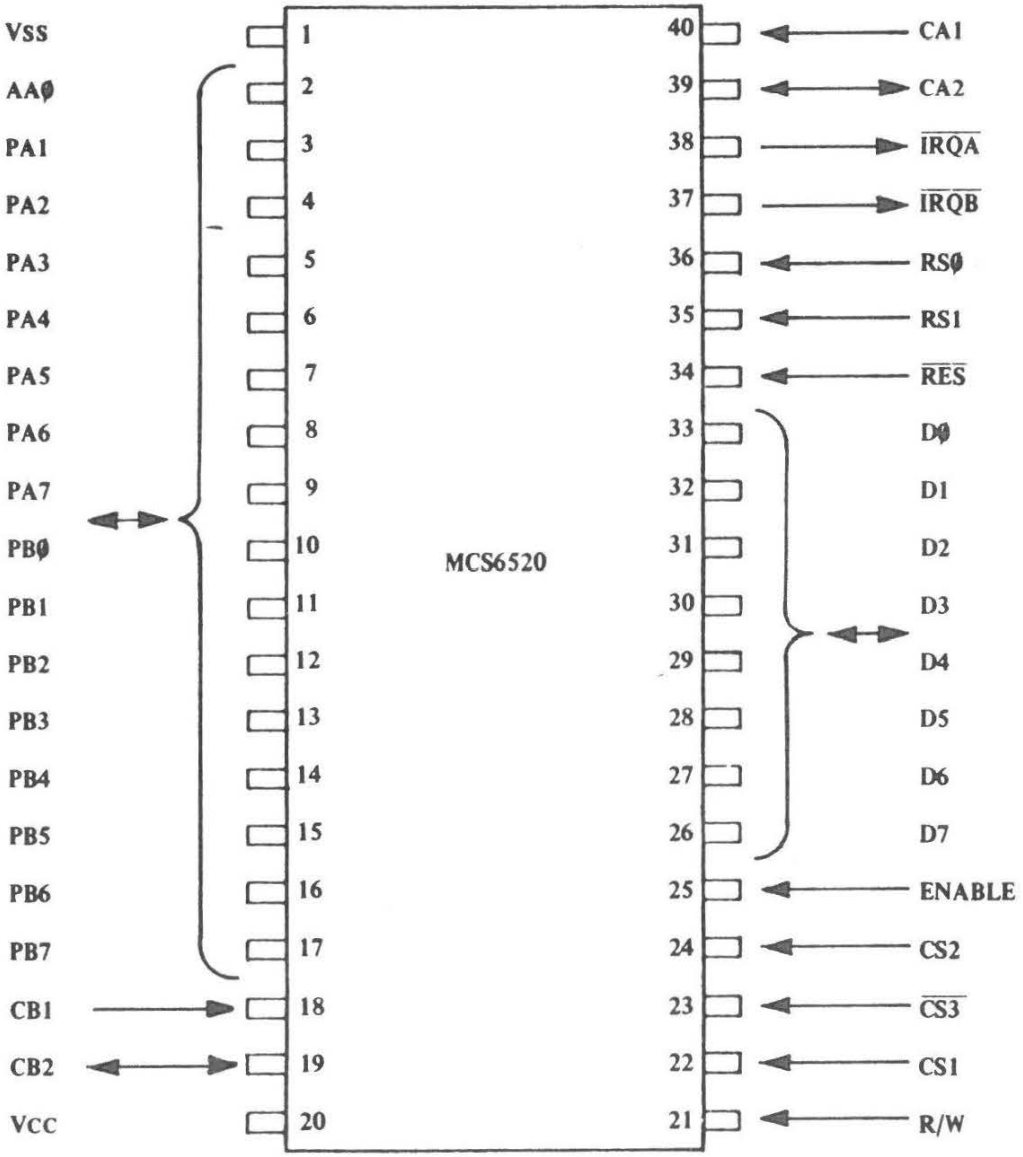
Die Konfiguration des MCS 6520 wird durch den Mikroprozessor programmiert, wenn das System auf seine Anfangsbedingungen gesetzt wird. Jede der peripheren Datenleitungen wird entweder als Eingang oder Ausgang programmiert. Jede der vier Steuerungs/Interrupt Leitungen kann für eine der vier möglichen Steuerungsarten programmiert werden. Dadurch wird ein hoher Grad an Flexibilität in der Gesamtwirkungsweise der Schnittstelle ermöglicht. Es folgen einige der wichtigeren Eigenschaften des MCS 6520:

- Kompatibilität mit den MCS 650X Mikroprozessoren.
- 8-Bit zweirichtungs Datenbus zur Verbindung mit dem Mikroprozessor.
- Zwei 8-Bit zweirichtungs Ports zur Verbindung mit Peripheriegeräten.
- Zwei programmierbare Steuerregister.
- Zwei programmierbare Datenrichtungs-Register.
- Zwei einzeln steuerbare Interrupt-Eingangsleitungen - zwei davon verwendbar als periphere Steuerausgänge.
- Handshake Steuerlogik für peripheren Eingangs- und Ausgangsbetrieb.
- Periphere Leitungen mit hoher Impedanz und drei Ausgangszuständen und direkter Transistor-Ansteuerung.
- Programmgesteuerter Interrupt und Interrupt Maskierungsmöglichkeit.

1.5.2 Organisation des MCS 6520

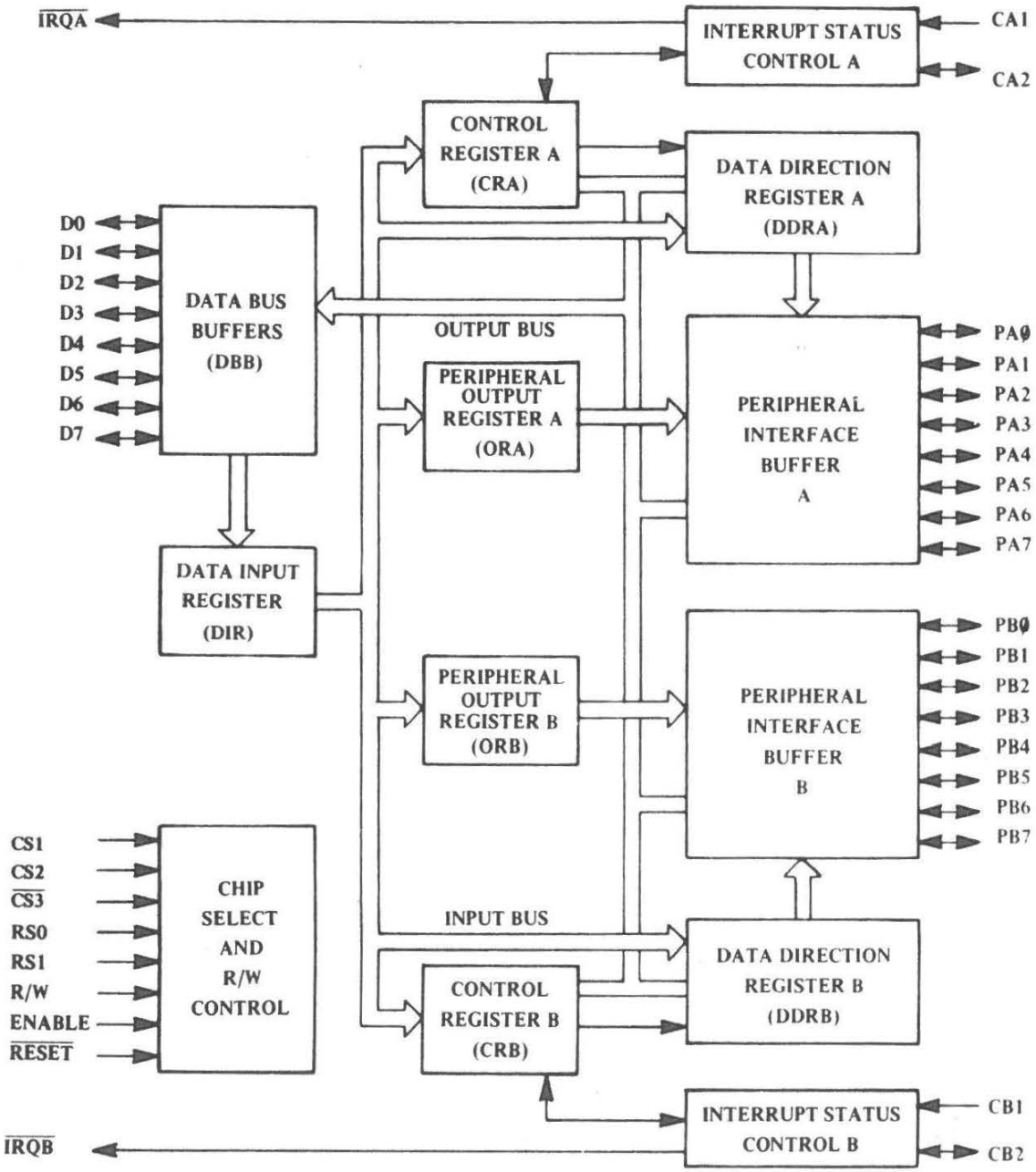
Das Blockschaltbild, Bild 1.25, zeigt die internen Register, Datenwege und die verschiedenen Ein- und Ausgänge des Bausteins MCS 6520. Dieser Abschnitt enthält eine allgemeine Beschreibung der inneren Organisation des Bausteins und eine Erläuterung, wie die verschiedenen Register aufeinander wirken. Die folgenden Abschnitte beschreiben die Wirkungsweise der Ein- und Ausgänge und die Arbeitsweise jedes einzelnen Registers. Der letzte Abschnitt bespricht den MCS 6520 auf die Anwendung bezogen. Die Wechselwirkung der Register Bits der Ein- und Ausgangsleitungen usw. wird beschrieben.

Der MCS 6520 ist in zwei unabhängigen Abschnitten organisiert, die als „ A - Seite " und „ B - Seite " bezeichnet werden. Jeder Abschnitt besteht aus einem Steuerregister (CRA, CRB), Datenrichtungsregister (DDRA, DDRB), Ausgangsregister (ORA, ORB), Interrupt Statussteuerung und dem Puffer zum Treiben der peripheren Schnittstellenbusse.



MCS6520 Pinout Designations Peripheral Interface Adaptor

ABBILDUNG 1.24: MCS 5420 STIFTBELEGUNG DES PERIPHEREN INTERFACE ADAPTERS



MCS6520 Internal Architecture

ABBILDUNG 1.25: MCS 6520 INNERER AUFBAU

1.5.2.1 Dateneingangs Register

Wenn der Mikroprozessor Daten in den MCS 6520 einschreibt, werden die Daten auf dem Datenbus während des Phase 2 Taktes im Dateneingangsregister gespeichert. Nach der abfallenden Flanke der Phase 2 werden diese Daten in eines der sechs internen Register übertragen. Dadurch wird sichergestellt, daß die Ausgangslinien saubere Übergänge von hohen zu niedrigem oder von niedrigem zu hohen Pegeln machen und die Spannung stabil bleibt, außer wenn sie auf die entgegengesetzte Polarität wechselt.

1.5.2.2. Steuerregister (CRA und CRB)

Bild 1.29 zeigt die Bit-Bezeichnungen und Funktionen in den Steuerregistern. Mit den Steuerregistern kann der Mikroprozessor die Wirkungsweise der Interrupt-Leitungen (CA1, CA2, CB1,CB2,)und die der peripheren Steuerleitungen (CA2, CB2,) steuern. Ein einzelnes Bit in jedem Register steuert die Adressierung der Datenrichtungsregister (DDRA, DDRB) und der Ausgangsregister (ORA, ORB,), die unten besprochen werden. Zusätzlich sind in jedem Steuerregister zwei Bit (Bit 6 und 7) vorgesehen, die den Status der Interrupt-Eingangsleitungen anzeigen (CA1, CA2, CB1, CB2). Diese Interrupt-Status-Bits ($\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$) werden normalerweise während des Programms zur Interrupt Bedienung abgefragt, um die Quelle eines aktiven Interrupts festzustellen. Dies sind die Interrupt-Leitungen, die den Interrupt-Eingang ($\overline{\text{IRQ}}$, $\overline{\text{NMI}}$) des Mikroprozessors treiben. Die anderen Bits im CRA und CRB sind in der Besprechung der Schnittstelle zum Peripherie-Baustein (Abschn. 1.5.4) besprochen.

1.5.2.3 Datenrichtungs-Register(DDRA, DDRB,)

Mit den Datenrichtungs-Registern kann der Prozessor jeden Anschluß eines 8 Bit Peripherie-Eingangs/Ausgangs-Kanals als entweder Eingang oder Ausgangs programmieren. Jedes Bit im DDRA steuert den entsprechenden Anschluß im Peripherie-Kanal A und jedes Bit im DDRB im Peripherie-Kanal B. Eine „ 0 " im Datenrichtungs-Register programmiert die entsprechende Peripherie-Leitung als Eingang, eines „ 1 " als Ausgangs.

Die Datenrichtungs-Register werden normalerweise nur während der Routine zur System-Initialisierung progeammiert, die als Antwort auf ein Rücksetz-Signal erfolgt. Der Inhalt dieser Register kann jedoch auch während des Systembetriebs geändert werden. Auf diese Weise können manche peripheren Geräte wie z.B. Tastaturen sehr gut gesteuert werden.

1.5.2.4 Periphere Ausgangs-Register (ORA, ORB)

Die peripheren Ausgangs-Register speichern die Ausgangsdaten, die am peripheren Ausgangskanal erscheinen. Eine in ein Bit im ORA eingeschriebene „ 0 " läßt die entsprechende Leitung im Peripherieausgangskanal A auf L gehen, wenn diese Leitung als Ausgangs programmiert ist. Eine „ 1 " läßt den Ausgangs auf H gehen. Auf die gleiche Weise werden die Leitungen des Peripheriekanals B gesteuert.

Die Adressierung dieser Register wird in Abschnitt 1.5.3.4 besprochen.

1.5.2.5 Steuerung des Interrupt-Status

Die Steuerung für den Interrupt-Status (A,B) steuert die vier Anschlüsse für Interrupt/Periphere Steuerung (CA1, CA2, CB1, CB2). Diese Logik wertet den Inhalt der entsprechenden Steuer-Register aus, erkennt aktive Flanken an den Interrupt-Eingängen und führt die Operationen aus, die zum sicheren Betrieb dieser vier peripheren Schnittstellen-Leitungen nötig sind. Der Betrieb dieser Leitungen wird in Abschnitt 1.5.4.2 im einzelnen beschrieben.

1.5.2.6 Puffer für periphere Schnittstelle (A, B) und Datenbus-Puffer (DBB)

Die Puffer liefern die Ströme und Spannungen, die zum Treiben der peripheren Eingangs/Ausgangs-Kanäle und des Datenbus nötig sind.

1.5.3 Schnittstelle zwischen MCS 6520 und MCS 650X

Die Verbindung zwischen dem MCS 6520 und dem Prozessor besteht aus einem 8 Bit Datenbus für zwei Richtungen und folgenden Leitungen: 3 zur Baustein-Auswahl 2 zur Register-Auswahl 2 zur Anforderung eines Interrupt, einer Schreib/Lese-Leitung, einer Einschalt (enable) - und einer Rücksetzleitung.

1.5.3.1 Datenbus (D0-D7)

Der 8 Bit Datenbus für zwei Richtungen dient zur Übertragung von Daten zwischen Prozessor und dem MCS 6520. Die Ausgangstreiber für den Datenbus haben drei Ausgangszustände. Sie bleiben im hochohmigen Zustand außer wenn der Prozessor Daten vom Peripherie-Adapter liest. Dieser Datenbus ist derselbe wie der in Abschnitt 1,3,1 besprochene.

1.5.3.2 Einschalt-Eingang (E, Enable)

Der Eingang E ist der einzige Eingang für den Zeitablauf zwischen Prozessor und peripherem Schnittstellen-Baustein. Er steuert alle Datenübertragungen von und zum MCS 6520. Im Normalbetrieb wird dieser Eingang mit dem Takt 2 verbunden. Der E-Impuls wird Takt 2 für den Ausgang.

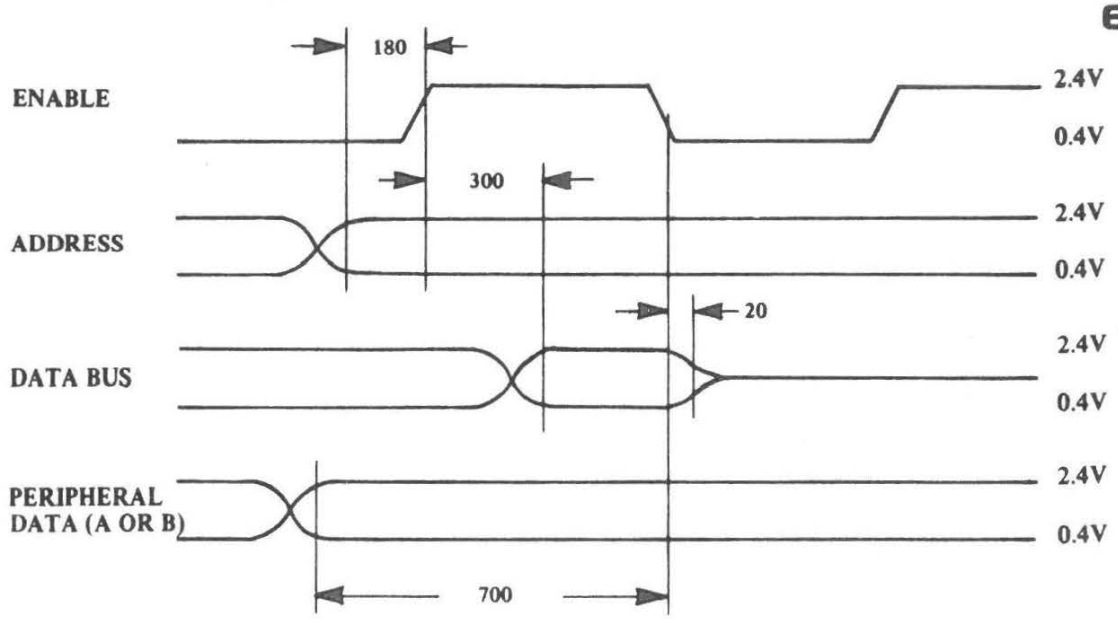
1.5.3.3 Schreiben/Lesen (R/W, Read/Write)

Mit diesem Signal steuert der Prozessor die Richtung der Datenübertragung auf dem Datenbus. Ein L auf dieser Leitung schaltet die Eingangs-Puffer ein (Prozessor schreibt). Der Eingang E steuert die Datenübertragung vom Prozessor zum MCS 6520, wenn der Baustein ausgewählt ist. Ein H auf der Leitung R/W läßt den MCS 6520 Daten auf die Datenbus-Puffer übertragen. Die Datenbus-Puffer werden eingeschaltet, wenn die richtigen Bausteinauswahl- und Einschalt-Signale anliegen.Bild 1.26 zeigt das Zeitdiagramm für Schreiben/Lesen.

1.5.3.4 Leitungen zur Bausteinauswahl (CS1, CS2, $\overline{CS3}$)

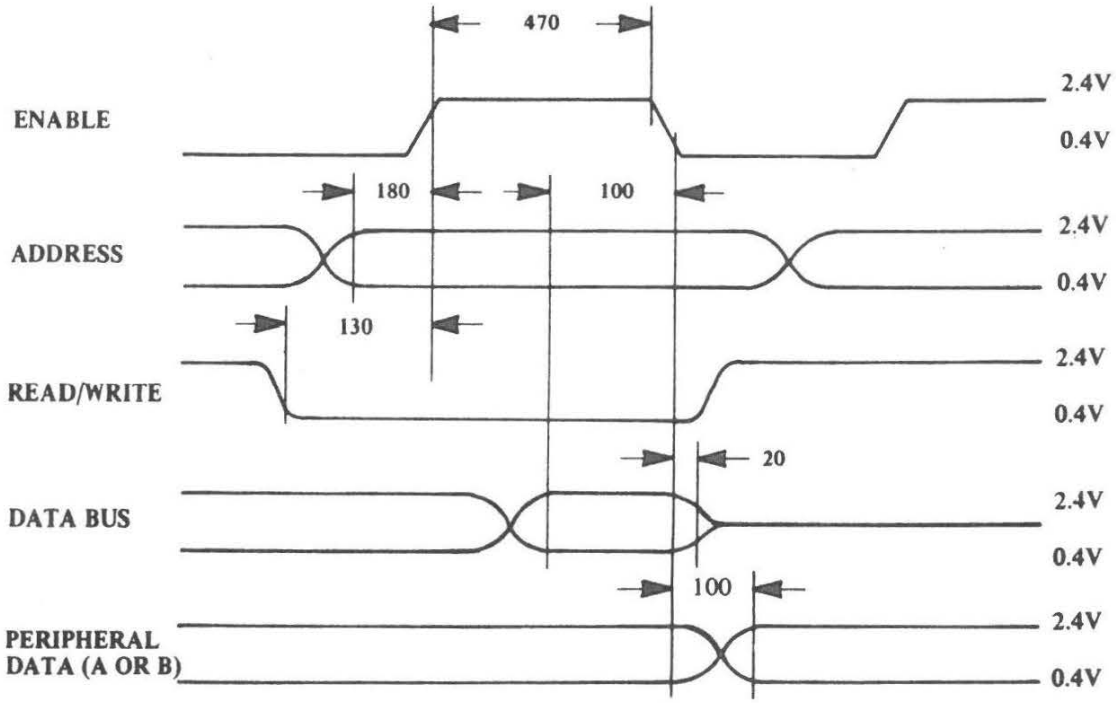
Mit diesen drei Eingängen kann der Prozessor den richtigen peripheren Schnittstellen-Baustein auswählen. Zur Auswahl des Bausteins müssen CS1 und CS2 auf H und $\overline{CS3}$ auf L liegen. Die Signale E und R/W steuern dann die Datenübertragung. Diese Leitungen sind normalerweise mit Adressleitungen des Prozessors verbunden, entweder direkt oder durch Adressdekoder.

Wie in Abschnitt 1.5.5.2 beschrieben, steuert ein einzelnes Bit in jedem Steuer-Register (CRA und CRA) den Zugriff zum Datenrichtungs-Register oder der peripheren Schnittstelle. Wenn Bit 2 im Steuerregister „ 1 " ist, wird ein peripheres Ausgangs-Register (ORA, ORB,) ausgewählt, und wenn Bit 2 „ 0 " ist, wird das Datenrichtungs-Register ausgewählt.



Microprocessor Interface Timing - Read

ABBILDUNG 1.26A: MIKROPROZESSOR ZEITDIAGRAMM LESEN



***NOTE: ALL TIMES SPECIFIED ARE IN nSEC FOR 1MHZ OPERATION.**

Microprocessor Interface Timing - Write

ABBILDUNG 1.26B: MIKROPROZESSOR ZEITDIAGRAMM SCHREIBEN

Die internen Register werden durch die Register-Auswahlleitungen (RS0, RS1,) und das Bit für den Zugriff zum Datenrichtungs-Register wie folgt gesteuert:

		Steuer-Bit für den Zugriff zum Datenrichtungs-Register		
RS 1	RS 0	CRA-2	CRB-2	ausgewähltes Register
0	0	1	-	periphere Schnittstelle A (siehe Abschnitt 1.5.3.5.1)
0	0	0	-	Datenrichtungs-Register A
0	1	-	-	Steuer-Register A
1	0	-	1	periphere Schnittstelle B (siehe Abschnitt 1.5.3.5.2)
1	0	-	0	Datenrichtungs-Register B
1	1	-	-	Steuer-Register B

Wenn der Programmierer Daten in DDRA, ORA, DDRB, oder ORB einschreiben will, muß er zuerst Bit 2 des entsprechenden Steuer-Registers setzen. Dann kann auf das gewünschte Register zugegriffen werden. Die dafür nötige Adresse wird durch das verwendete Adress-Verbindungsverfahren festgelegt (in Kapitel 3, Abschnitt 2.3.1 ist die Adressierung in MCS 650X-Systemen besprochen).

1.5.3.5 Register-Auswahlleitungen (RS0, RS1)

Diese zwei Leitungen dienen zur Auswahl der verschiedenen Register im MCS 6520. Diese Eingangsleitungen wählen zusammen mit internen Steuer-Registern ein bestimmtes Register zum Zugriff durch den Prozessor aus. Normalerweise sind diese Leitungen mit den Adress-Ausgangsleitungen des Prozessors verbunden.

Diese Leitungen zusammen mit den Eingängen zur Bausteinauswahl erlauben dem Prozessor die Adressierung eines einzelnen 8 Bit Registers in seinem Adressbereich. Dies Register kann ein internes Register (CRA, ORA usw) oder ein peripherer Eingangs/Ausgangs-Kanal sein.

Der Prozessor kann direkt in die Steuer-Register (CRA,CRB), die Datenrichtungs-Register (DDRA, DDRB) und die Peripherieausgangs-Register (ORA, ORB) einschreiben. Zusätzlich kann der Prozessor den Inhalt der Steuer-Register und der Datenrichtungs-Register direkt lesen. Der Zugriff zum Peripherieausgangs-Register zum Auslesen von Daten in den Prozessor geht anders als bei den Registern ORA und ORB vor sich. Er wird deshalb weiter unten getrennt besprochen.

1.5.3.5.1 Auslesen des peripheren Eingangs/Ausgangs-Kanals A

Der periphere Eingangs/Ausgangs-Kanal A besteht aus acht Leitungen, die als Eingänge oder Ausgänge programmiert werden können. Als Ausgang programmiert, zeigt jede Leitung den Inhalt des entsprechenden Bit im peripheren Ausgangs-Register an. Wenn sie als Eingang programmiert sind, gehen diese Leitungen, abhängig von den Eingangsdaten, auf H oder L. Das periphere Ausgangs-Register (ORA,) wirkt nicht auf die Leitungen, die als Eingänge programmiert sind. Bild 1.27a zeigt das Zeitdiagramm der Schnittstelle.

Ein Lesevorgang überträgt die Daten auf den peripheren Eingangs/Ausgangsleitungen A direkt auf den Prozessor (über den Datenbus), wenn RS1 = 0,RS0 = 0 und CRA 2 = 1 ist. Der Bus enthält dann sowohl Eingangs- als auch Ausgangsdaten. Der Prozessor muß so programmiert sein, daß er nur diejenigen Bits erkennt und auswertet, die für den bestimmten peripheren Vorgang nötig sind.

Der Prozessor liest die Anschlüsse des peripheren Eingangs/Ausgangs-Kanal A statt des peripheren Ausgangs-Registers (ORA). Es ist daher möglich, daß die in den Prozessor eingelesenen Daten für eine Ausgangsleitung vom Inhalt des peripheren Ausgangs-Registers abweichen. Das trifft zu, wenn der Eingangs/Ausgangs-Anschluß nicht bis auf 2,4 V gehen kann, und das periphere Ausgangs-Register eine logische 1 enthält. In diesem Fall liest der Prozessor eine logische 0 aus dem Peripherieanschluß A, selbst wenn das entsprechende Bit im peripheren Ausgangs-Register eine logische 1 enthält.

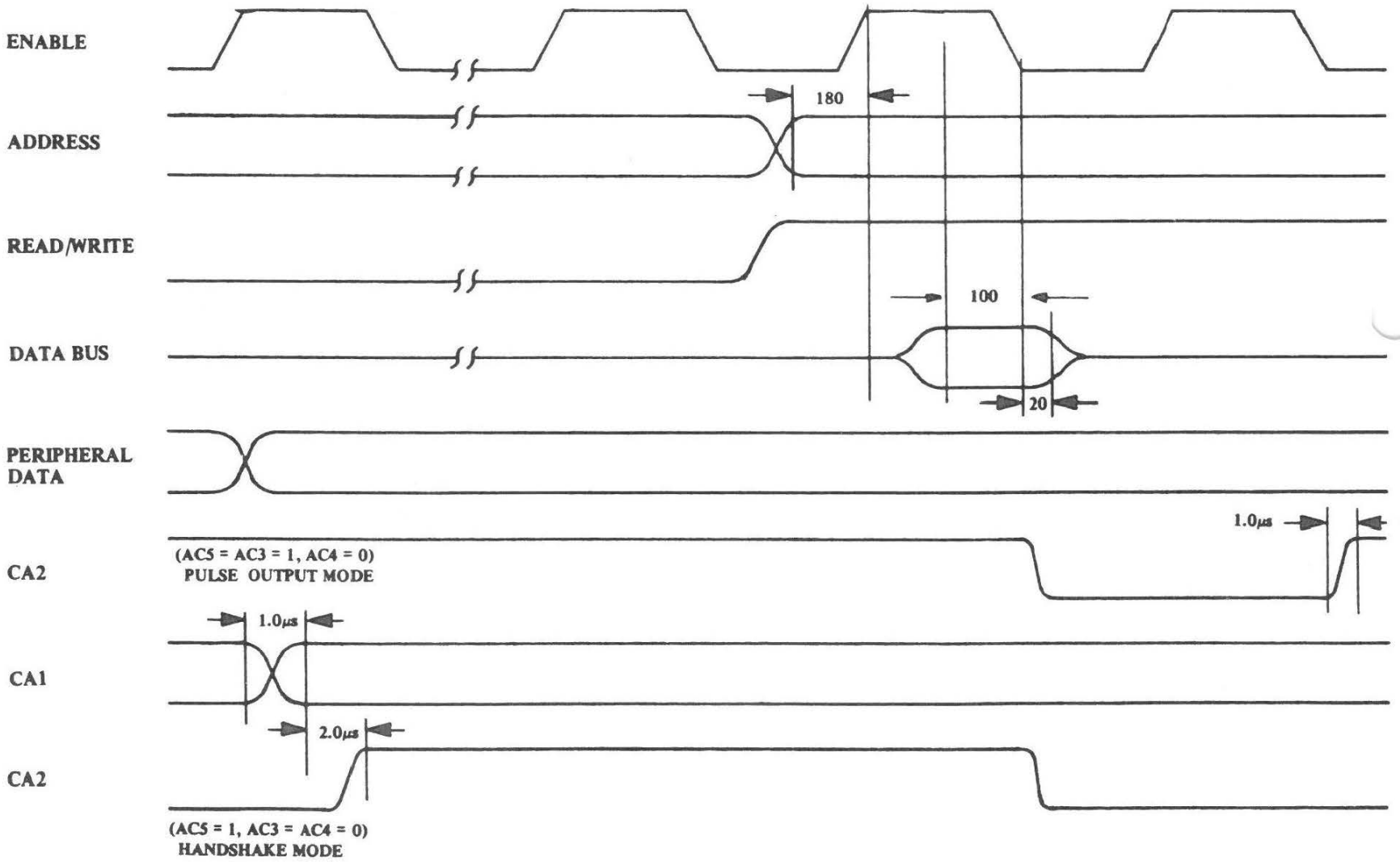
1.5.3.5.2 Auslesen des peripheren Eingangs/Ausgangs-Kanals B

Das Auslesen ergibt, ähnlich wie beim peripheren Kanal A, eine Kombination von Eingangs- und Ausgangsdaten. Die Daten vom peripheren Ausgangs-Register B (ORB) werden jedoch direkt ausgelesen, wenn diese Leitungen als Ausgänge programmiert sind. Man kann daher die Ausgangsleitungen der Peripherie B stark bealsten, ohne daß bei einem Lesevorgang falsche Daten in den Prozessor zurückübertragen werden. Bild 1.27b zeigt das Zeitdiagramm.

Die Einzelheiten des Peripherie-Kanals A und B werden im nächsten Abschnitt besprochen.

1.5.3.6 Rücksetzen ($\overline{\text{RES}}$, Reset)

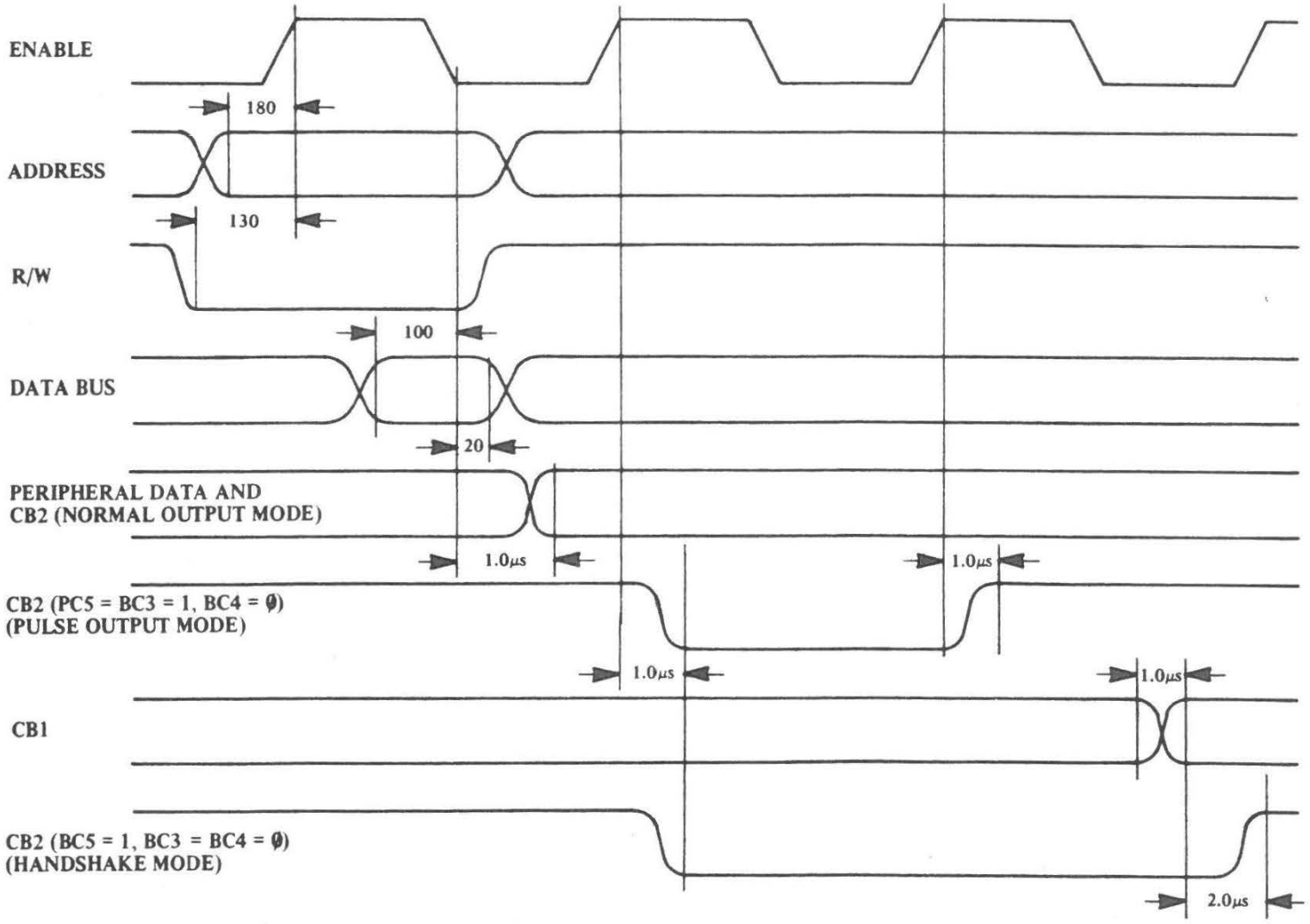
Diese Leitung setzt den Inhalt aller Register im MCS 6520 auf logisch 0, wenn sie auf L liegt. Sie kann zum Rücksetzen beim Einschalten oder zum allgemeinen Rücksetzen während des Systembetriebes dienen.



NOTE: ALL TIMES SPECIFIED ARE IN nSEC FOR 1MHZ OPERATION.

Peripheral A Interface Timing

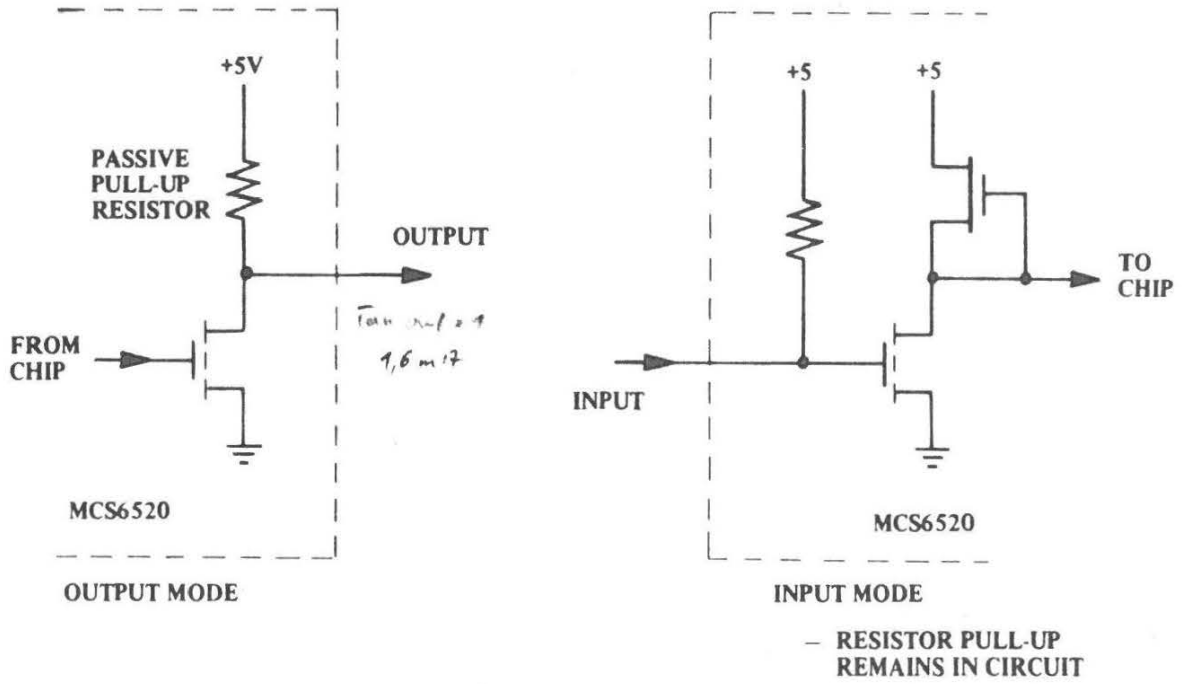
ABBILDUNG 1.27A: PERIPHERIE A - ZEITDIAGRAMM



NOTE: ALL TIMES SPECIFIED ARE IN nSEC FOR 1MHZ OPERATION.

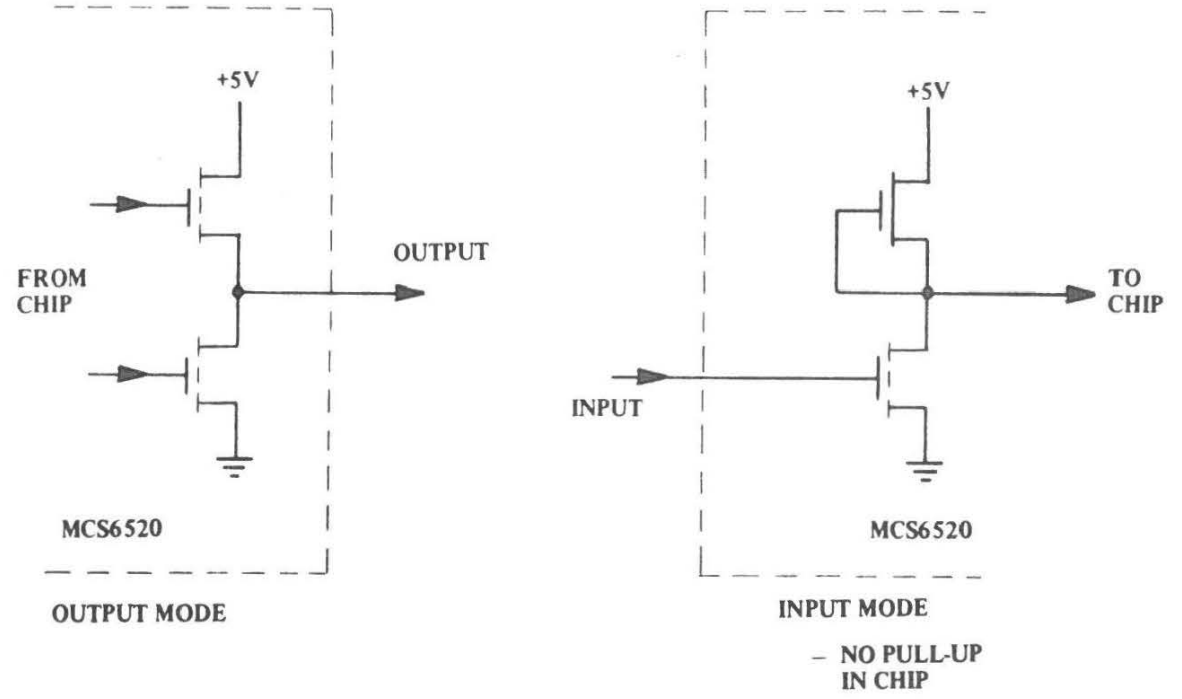
Peripheral B Interface Timing

ABBILDUNG 1.27B: PERIPHERIE B - ZEITDIAGRAMM



Peripheral I/O Port A Buffer

ABBILDUNG 1.28A: PERIPHERIE E/A-KANAL A PUFFER



Peripheral I/O Port B Buffer

ABBILDUNG 1.28B: PERIPHERIE E/A-KANAL B PUFFER

1.5.3.7 Anforderungs-Leitung für Interrupt ($\overline{IRQA}$, $\overline{IRQB}$)

Ein L auf den Anforderungs-Leitungen für Interrupt ($\overline{IRQA}$ und $\overline{IRQB}$) gibt dem Prozessor entweder direkt oder durch eine externe Schaltung für die Interrupt-Priorität einen Interrupt. Diese Ausgänge können 1,6 mA von einer externen Quelle aufnehmen. Sie haben kein internes Bauelement gegen V_{DD}. Daher können alle „Anforderungs-Leitungen für den Interrupt als „ verdrahtetes ODER" verbunden werden. A und B bezeichnen den zugehörigen Peripherie-Kanal.

Jede Anforderungs-Leitung hat zwei Bits, die die Leitung auf L setzen können. Es sind die Bits 6 und 7 in den zwei Steuer-Registern. Sie dienen als Verbindung zwischen den peripheren Interrupt-Signalen und den Interrupt-Eingängen des Prozessors. Jedes Bit hat ein entsprechendes Bit zur Abschaltung des Interrupts, das dem Prozessor das Ein- oder Ausschalten des Interrupts von jedem der vier Interrupt-Eingänge (CA1, CA2, CB1, CB2,) erlaubt.

Die vier Interrupt-Bits werden durch aktive Flanken des Signals am Interrupt-Eingang (CA1, CA2, CB1, CB2) gesetzt. Die Steuerung dieser aktiven Flanke wird im nächsten Abschnitt bei der Besprechung der Schnittstelle zwischen dem MCS 6520 und dem Peripheriegerät erläutert.

1.5.3.7.1 Steuerung von $\overline{IRQA}$

Bit 7 des Steuer-Registers A wird immer durch eine aktive Flanke des Signals am Interrupt-Eingang CA1 gesetzt. Der Interrupt von diesem Bit kann abgeschaltet werden, indem man Bit 0 im Steuer-Register A (CRA) auf logisch 0 setzt. Ebenso wird Bit 6 des Steuer-Registers A durch eine aktive Flanke des Signals am Interrupt-Eingang CA2 gesetzt. Der Interrupt von diesem Bit kann abgeschaltet werden, indem Bit 3 im Steuer-Register auf logisch 0 gesetzt wird.

Bit 6 und Bit 7 in CRA werden durch Auslesen des peripheren Ausgangs-Registers A (Read Peripheral Output Register A) zurückgesetzt. Zu diesem Vorgang müssen die richtigen Signale zur Auswahl des Bausteins und des Registers vorliegen, damit der Prozessor den peripheren Eingangs/Ausgangs-Kanal A auslesen kann.

1.5.3.7.2 Steuerung von $\overline{\text{IRQB}}$

$\overline{\text{IRQB}}$ wird auf die gleiche Weise wie $\overline{\text{IRQA}}$ gesteuert. Bit 7 im CRB wird durch eine aktive Flanke auf CB1 gesetzt, der Interrupt von diesem Bit wird von Bit 0 des CRB gesteuert. Bit 6 des CRB wird durch eine aktive Flanke von CB2 gesetzt, der Interrupt von diesem Bit wird durch Bit 3 des CRB gesteuert. Durch Auslesen des peripheren Ausgangs-Registers B werden Bit 6 und Bit 7 zurückgesetzt.

Zusammenfassung:

IRQA geht auf L, wenn CRA-7 = 1 und CRA-0 = 1 oder wenn CRA-6 = 1 und CRA-3 = 1 ist.

IRQB geht auf L wenn CRB-7 = 1 und CRB-0 = 1 oder wenn CRB-6 = 1 und CRB-3 = 1 ist.

Der Gebrauch dieser Interrupt-Bits und Interrupt-Abschalt-Bits wird in Abschnitt 1.5.4 genauer besprochen.

Es wird betont, daß die Bits Bytes Verbindung zwischen den peripheren Interrupt-Signalen und den Interrupt-Eingängen des Prozessors dienen. Der Prozessor steuert den Interrupt mit den Abschalt-Bits für den Interrupt.

1.5.4 Schnittstelle zwischen MCS 6520 und Peripherie-Geräten

Der MCS 6520 hat für die Schnittstelle mit Peripherie-Geräten zwei 8 Bit Kanäle für zwei Richtungen und vier Leitungen für Interrupt/Steuerung. Diese Kanäle und die damit verbundenen Leitungen für Interrupt/Steuerung werden als Seite A und Seite B bezeichnet. Jede Seite hat ihre besonderen Eigenschaften und wird deshalb unten getrennt besprochen.

1.5.4.1 Periphere Eingangs/Ausgangs-Kanäle

Über die peripheren Eingangs/Ausgangs-Kanäle A und B hat der Prozessor mit den Eingangsleitungen des Peripherie-Gerätes Verbindung, in dem er Daten in das periphere Ausgangs-Register lädt. Die Verbindung mit den Ausgangsleitungen des Peripherie-Gerätes wird durch direktes Lesen der Daten der Eingangs-Leitungen des Peripherie-Kanales auf dem Datenbus und in die internen Register des Prozessors bewirkt.

1.5.4.1.1 Peripherer Eingangs/Ausgangs-Kanal A (PA0-PA7)

Wie in Abschnitt 1.5.2.3 besprochen, kann jede der peripheren Eingangs/Ausgangs-Leitungen als Eingang oder Ausgang programmiert werden. Dies wird durch eine 1 im entsprechenden Bit des Datenrichtungs-Registers für einen Ausgang, oder durch eine 0 für einen Eingang erreicht.

Die Treiber für die Leitungen haben einen Lastwiederstand, wie in Bild 1.28a gezeigt. Die Ausgangsspannung wird deshalb gegen Vdd für eine logische 1 gezogen. Das aktive Bauelement kann 1,6 mA aufnehmen, und damit eine Standard-TTL-Last treiben. Beim Betrieb als Eingang sind die in Bild 1.28a gezeigten Bauelemente noch mit dem Eingangs/Ausgangs-Anschluß verbunden und liefern Strom in diesen Anschluß. Aus diesem Grund stellen diese Anschlüsse im Betrieb als Eingang eine Standard-TTL-Last dar.

1.5.4.1.2 Peripherer Eingangs/Ausgangs-Kanal B (PBØ-PB7)

Der Kanal B weicht nur in seinen Ausgangs-Puffern vom Kanal A ab.

Die Puffer der peripheren Eingangs/Ausgangs-Kanals B haben einen Gegentakt-Ausgang, Bild 1.28b. Die Bauelemente gegen Vdd sind für eine logische 0 abgeschaltet und für eine logische 1 angeschaltet. Es wird nicht garantiert, daß die Spannung für logisch 1 höher als + 2,4 V ist, da es sich um aktive Bauelemente handelt. Die Ausgänge sind mit TTL-kompatibel aber nicht mit CMOS.

Bei 1,5 V können die aktiven Bauelement gegen Vdd ein mA aufnehmen. Damit können direkt Schalter mit Darlington-Transistoren getrieben werden. So ist eine sehr einfache Steuerung von Relais, Lampen usw. möglich.

Für die als Eingang programmierten Leitungen werden die Ausgangsdaten direkt vom peripheren Ausgangs-Register gelesen, da diese Ausgänge zum direkten Treiben von Transistoren vorgesehen sind.

Die Gegentakt-Puffer des Peripherie-Kanals B haben einen hochohmigen Eingangszustand. Der Ausgangspuffer geht in den hochohmigen Zustand, wenn die betreffende Eingangs/Ausgangs-Leitung als Eingang programmiert ist. Dann beträgt die Impedanz mehr als 1 MOHM.

1.5.4.2 Interrupt-Eingang/Peripherie-Steuerung (CA1,CA2, CB1,CB2)

Diese vier Leitungen bieten eine Zahl von besonderen Funktionen zur Steuerung der Peripherie. Sie vergrößern die Anwendbarkeit der beiden Peripherie-Schnittstellen-Kanäle (PA0-PA7, PB0-PB7).

1.5.4.2.1 Peripherie A: Interrupt-Eingang/Peripherie-Steuerung (CA1, CA2)

CA ist ausschließlich ein Interrupt-Eingang. Eine aktive Flanke an diesem Eingang setzt Bit 7 von CRA auf logisch 1. Der Prozessor kann die aktive Flanke programmieren. Das Interrupt-Bit (Bit 7 des CRA) wird durch eine negative Flanke des Signals an CA1 gesetzt, wenn in Bit 1 des CRA eine 0 steht, und durch die positive Flanke gesetzt, wenn eine 1 dort steht. Eine negative Flanke ist der Übergang von H zu L, eine positive der Übergang von L zu H.

Der Prozessor bekommt über IRQA einen Interrupt, wenn das Interrupt-Bit gesetzt ist und Bit 0 des CRA eine 1 enthält, wie vorher beschrieben.

CA 2 kann als vollkommen unabhängiger Interrupt-Eingang oder als Ausgang zur Peripherie-Steuerung dienen. Als Eingang (CRA, Bit 5 = 0) setzt er das Interrupt-Bit, Bit 6 des CRA, bei der durch Bit 4 des CRA ausgewählten aktiven Flanke auf logisch 1.

Diese Steuer-Register-Bits und Interrupt-Eingänge dienen demselben Zweck wie oben für CA 1 beschrieben. Das Eingangssignal setzt das Interrupt-Bit, das als Verbindung zwischen dem Peripherie-Gerät und der Interrupt-Struktur des Prozessors dient. Mit dem Abschalt-Bit für den Interrupt kann der Prozessor die System-Interrupts steuern.

In der Betriebsart Ausgang (CRA, Bit 5 = 1) kann CA 2 unabhängig einen einfachen Impuls erzeugen, wenn der Prozessor die Daten am peripheren Eingangs/Ausgangs-Kanal A liest. Für diese Betriebsart werden im CRA Bit 4 auf 0 und Bit auf 1 gesetzt. Dieser Impuls-Ausgang kann zur Steuerung der Zähler, Schiebe-Register usw. verwendet werden, die sequentielle Daten auf den peripheren Eingangs-Leitungen liefern.

In einer zweiten Betriebsart als Ausgang kann CA 2 in Verbindung mit CA 1 verwendet werden, um „ handshaking " zwischen dem Prozessor und dem Peripherie-Baustein auszuführen. Auf der Seite A erlaubt dies Verfahren die Steuerung der Datenübertragung vom Peripherie-Baustein in den Prozessor. Der Eingang CA 1 zeigt dem Prozessor die Verfügbarkeit von Daten durch einen Interrupt an. Der Prozessor liest die Daten und setzt CA 2 auf L. Damit wird dem Peripherie-Baustein angezeigt, daß er neue Daten liefern kann. Das Verfahren wird im Kapitel 2 besprochen.

Eine weitere Betriebsart als Ausgang kann ausgewählt werden, wenn Bit 4 des CRA auf 1 gesetzt wird. Hierbei ist CA 2 ein einfacher peripherer Steuer-Ausgang, der auf H oder L geht, wenn Bit 3 des CRA auf 1 oder 0 gesetzt wird. Die Arbeitsweise von CA 1 und CA 2 ist im nächsten Abschnitt zusammengefaßt.

1.5.4.2.2 Peripherie B: Interrupt-Eingang/Peripherie-Steuerung (CB 1, CB2,)

CB 1 arbeitet genau wie CA 1 nur als Interrupt-Eingang. Bit 7 von CRB wird durch die aktive Flanke gesetzt, die von Bit 0 des CRB ausgewählt ist. Beim Betrieb als Eingang arbeiten CB 2 und CA 2 genau gleich. Beim Betrieb als Ausgang (CRB, Bit 5 = 1) unterscheidet sich CB 2 etwas von CA 2. Es gibt einen Ausgangs-Impuls, wenn der Prozessor Daten in das periphere Ausgangs-Register B einschreibt. Das „ handshaking" findet für Datenübertragungen vom Prozessor zum Peripherie-Baustein statt.

Die Arbeitsweise von CB1 und CB2 wird im nächsten Abschnitt zusammengefaßt. Das „ handshaking" mit dem Peripherie-Kanal B wird in Kapitel 2 genauer besprochen.

	7	6	5 4 3	2	1 Ø
CRA	IRQA1	IRQA2	CA2 Control	DDRA Access	CA1 Control

	7	6	5 4 3	2	1 Ø
CRB	IRQB1	IRQB2	CB2 Control	DDRB Access	CB1 Control

Control Register Bit Designations

ABBILDUNG 1.29: STEUERREGISTER BIT-BESCHREIBUNG

CRA (CRB) Bit 1	CRA (CRB) Bit 0	Active Transition of Input Signal*	IRQA (IRQB) Interrupt Outputs
0	0	negative	Disable--remain high
0	1	negative	Enabled--goes low when bit 7 in CRA (CRB) is set by active transition of signal on CA1 (CB1)
1	0	positive	Disable--remain high
1	1	positive	Enable--as explained above

*Note 1: Bit 7 of CRA (CRB) will be set to a logic 1 by an active transition of the CA1 (CB1) signal. This is independent of the state of Bit 0 in CRA (CRB).

Control of Interrupt Inputs CA1, CB1

ABBILDUNG 1.30: STEUERUNG DER INTERRUPT-EINGÄNGE

CRA (CRB) Bit 5	CRA (CRB) Bit 4	CRA (CRB) Bit 3	Active Transition of Input Signal*	IRQA (IRQB) Interrupt Output
0	0	0	negative	Disable--remains high
0	0	1	negative	Enabled--goes low when bit 6 in CRA (CRB) is set by active transition of signal on CA2 (CB2)
0	1	0	positive	Disable--remains high
0	1	1	positive	Enable--as explained above

*Note: Bit 6 of CRA (CRB) will be set to a logic 1 by an active transition of the CA2 (CB2) signal. This is independent of the state of Bit 3 in CRA (CRB).

Control of CA2 (CB2) as Interrupt Inputs (Bit 5 = "0")

ABBILDUNG 1.31A: STEUERUNG VON CA 2 (CB 2) ALS INTERRUPT-EINGANG (BIT 5 = "0")

CRA Bit 5	CRA Bit 4	CRA Bit 3	Mode	Description
1	0	0	"Handshake" on Read	CA2 is set high on an active transition of the CA1 interrupt input signal and set low by a microprocessor "Read A Data" operation. This allows positive control of data transfers from the peripheral device to the microprocessor.
1	0	1	Pulse Output	CA2 goes low for one cycle after a "Read A Data" operation. This pulse can be used to signal the peripheral device that data was taken.
1	1	0	Manual Output	CA2 set low
1	1	1	Manual Output	CA2 set high

Control of CA2 Output Modes

ABBILDUNG 1.31B: STEUERUNG DER CA 2 AUSGANGS-ARTEN

CRB Bit 5	CRB Bit 4	CRB Bit 3	Mode	Description
1	0	0	"Handshake" on Write	CB2 is set low on microprocessor "Write B Data" operation and is set high by an active transition of the CB1 interrupt input signal. This allows positive control of data transfers from the microprocessor to the peripheral device.
1	0	1	Pulse Output	CB2 goes low for one cycle after a microprocessor "Write B Data" operation. This can be used to signal the peripheral device that data is available.
1	1	0	Manual Output	CB2 set low
1	1	1	Manual Output	CB2 set high

Control of CB2 Output Modes

ABBILDUNG 1.31C: STEUERUNG DER CB 2 AUSGANGS-ARTEN

1.5.5.2 Arbeitsweise des MCS 6520 in MCS 6500 Systemen

Ein kurzer Rückblick auf die Gesamtarbeitsweise des MCS 6520 dient zur Zusammenfassung der vielen vorher besprochenen Einzelheiten.

Während der Routine zur Herstellung der Anfangsbedingungen im System (Initialisierung), die als Antwort auf das Prozessor-Signal RESET ausgeführt wird, schreibt der Prozessor ein Muster von Einsen und Nullen in die Datenrichtungs-Register. Damit wird festgelegt, welche Leitungen als Eingänge und welche als Ausgänge arbeiten.

Dies Muster ist für den Systembetrieb normalerweise gleichbleibend. Der nächste Schritt wäre das Setzen der verschiedenen Betriebsarten, aktiven Flanken usw., die durch die Steuer-Register gesteuert werden. Zur gleichen Zeit kann das Steuer-Bit für den Zugriff zum Datenrichtungs-Register auf 1 gesetzt werden., um dem Prozessor die Steuerung der Peripheriekanäle während des Systembetriebs zu ermöglichen.

Die Interrupts bleiben normalerweise abgeschaltet, bis das gesamte System initialisiert ist. Dann werden die Interrupts eingeschaltet und der volle Betrieb des Systems beginnt.

Während des Systembetriebs fragt der Mikroprozessor die Schalter, Fühler usw. im Peripheriegerät durch Lesen der Daten auf den Peripherie-Eingangsleitungen ab. Auf die gleiche Art können binäre oder dezimale Daten in den Prozessor übertragen werden. Durch Einschreiben von Daten in die entsprechenden Bits der Periperieausgangs-Register werden die verschiedenen Lampen, Motore, Magnetspulen usw. im Peripheriegerät gesteuert. Die ganze Folge von Vorgängen wird durch den Programmierer bestimmt, um ein einzelnes Peripheriegerät auf festgelegte Weise zu steuern. Die verschiedenen Register, Gatter usw. im Schnittstellenbaustein dienen in erster Linie als Verbindung zwischen den inneren Vorgängen im Prozessor und den verschiedenen Ein- und Ausgängen in den gesteuerten Peripheriegeräten.

1.6 BAUSTEIN FÜR PERIPHERE SCHNITTSTELLE/SPEICHER - MCS 6530

1.6.1. Einführung

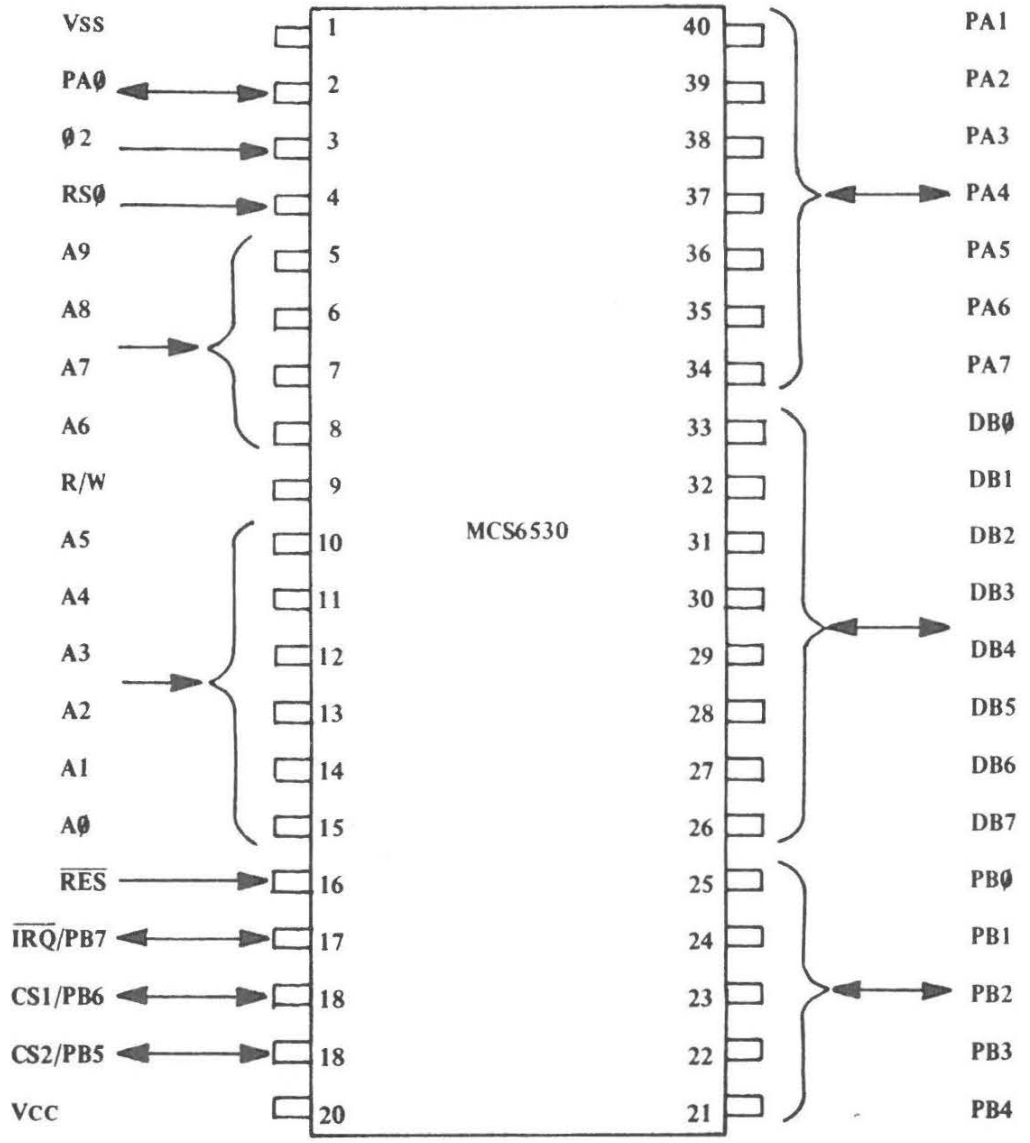
Der MCS 6530 ist zur Zusammenarbeit mit dem Mikroprozessor MCS 650X entwickelt worden. Er besteht aus einem Masken programmierbaren ROM 1024 x 8, einem RAM 64 x 8, zwei 8-Bit Kanälen für zwei Richtungen, die zur Verbindung zwischen Mikroprozessor und Peripheriegeräten dienen und einem programmierbaren Zeitgeber mit Interrupt, der verschiedene Zeiten von 1 bis 262.144 Taktperioden ausgibt.

Die Zusammenstellung des Eingangs/Ausgangs per Zeitgeber und die Interruptsmöglichkeiten werden von der Software gesteuert.

- 8 Bit Datenbus in zwei Richtungen zur Verbindung mit dem Mikroprozessor
- 2 8-Bit Kanäle in zwei Richtungen als direkte Schnittstelle zur Peripherie
- 2 Datenrichtungs-Register für peripheren Eingang/Ausgang
- Programmierbare Zeitgeber für 1 bis 256 x 1024 Taktperioden
- Programmierbarer Interrupt mit Zeitgeber
- CMOS - Kompatible Peripherie-Leitungen
- Peripherie-Anschlüsse zum direkten Ansteuern von Transistoren
- Datenanschlüsse mit drei Ausgangszuständen (Three-State)
- Bis zu 7 K zusammenhängendes ROM ohne externe Dekodierung
- ROM 1024 x 8
- Statisches RAM 64 x 8

1.6.2. Anschlußbelegung

Bild 1.33 zeigt die Anschlußbelegung des MCS 6530.



MCS6530 Pinout Designation

ABBILDUNG 1.32: MCS 6530 BESCHREIBUNG DER STIFTBELEGUNG

1.6.2.1. Rücksetzen ($\overline{RES}$)

Während der Initialisierung des Systems werden durch eine logische 0 am Eingang $\overline{RES}$ alle Eingangs/Ausgangs-Register 0 gesetzt. Während des Rücksetzens werden die Datenbus-puffer abgeschaltet. Der Interrupt ist während des Rücksetzens abgeschaltet. Das Signal $\overline{RES}$ muß für mindestens eine Taktperiode auf L gehalten werden, wenn Rücksetzung erfolgen soll.

1.6.2.2 Eingangstakt

Der Eingangstakt ist der Takt 2 des Systems. Er kann entweder einen niedrigen Pegel (V_{IL} 0,4 V, V_{IH} 2,4 V) oder einen hohen Pegel (V_{IL} 0,2 V, $V_{IH} = V_{CC}$ + 0,3 -0,2) haben.

1.6.2.3 Schreiben/Lesen (R/W)

Der Prozessor gibt das Signal R/W aus und benutzt es zur Steuerung der Datenübertragung zwischen Prozessor und dem MCS 6530. Ein H am Anschluß R/W erlaubt dem Prozessor (mit der richtigen Adresse) das Auslesen der vom MCS 6530 gelieferten Daten. Ein L erlaubt ein Einschreiben (mit der richtigen Adresse) in den MCS 6530.

1.6.2.4. Anforderung eines Interrupts ($\overline{IRQ}$)

Der Anschluß $\overline{IRQ}$ ist ein Interrupt-Anschluß vom Zeitgeber. Dieser Anschluß kann, wenn er nicht als Interrupt benutzt wird, als peripherer Eingangs/Ausgangs-Anschluß (PE 7) benutzt werden. Bei Benutzung als Interrupt muß er durch das Datenrichtungs-Register als Eingang gesetzt werden. Der Anschluß liegt normal H. L zeigt einen Interrupt vom MCS 6530 an. Ein externer Widerstand gegen V_{CC} ist nicht nötig. Der interne Widerstand kann jedoch durch eine Maskenänderung weggelassen werden, um ein wired-Or mit anderen Bausteinen zu ermöglichen.

1.6.2.5. Datenbus (D 0-D7)

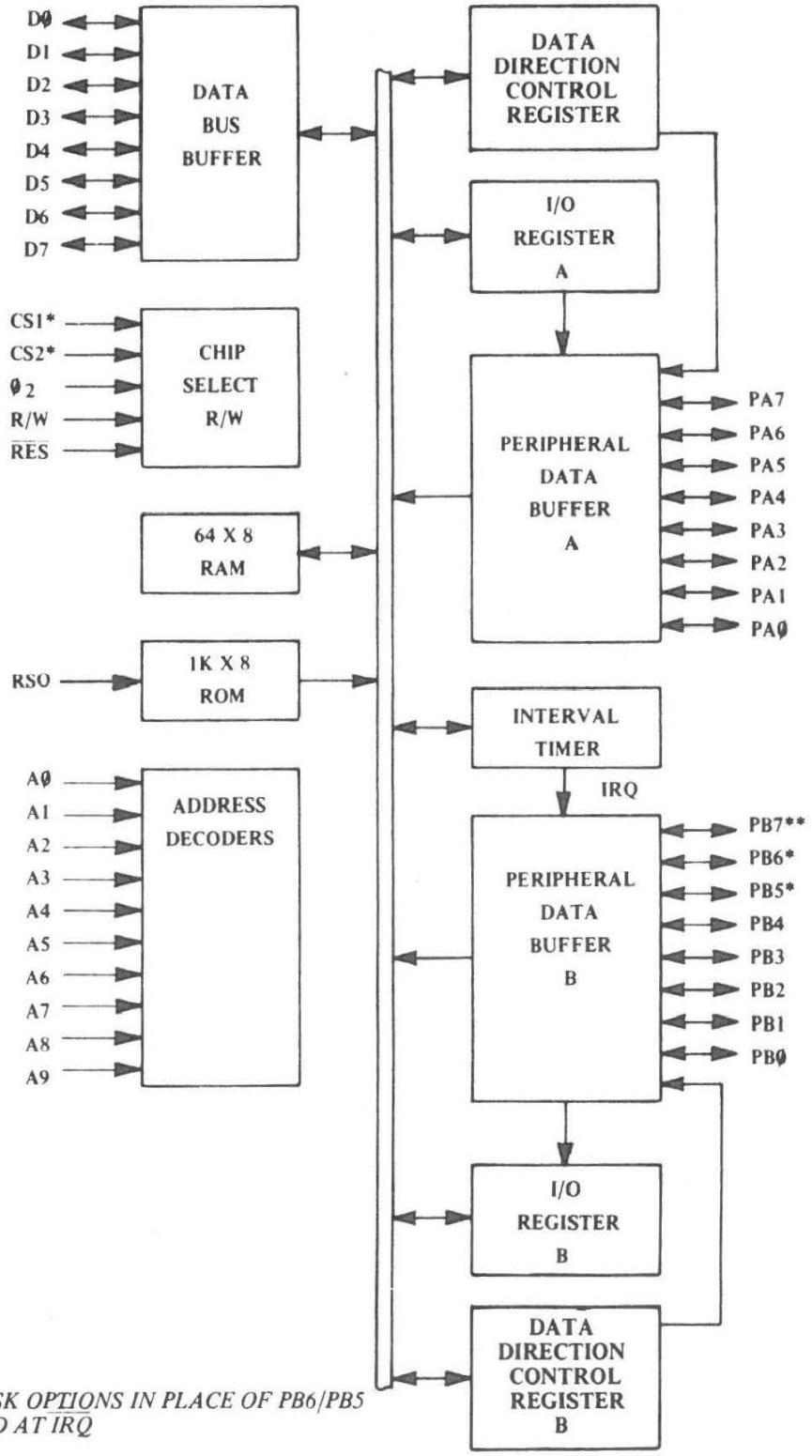
Der MCS 6530 hat 8 Datenanschlüsse für zwei Richtungen (DO - D7). Diese Anschlüsse gehen an die Datenleitungen des Systems und erlauben die Obertragung von Daten von und zum Prozessor. Die Ausgangs-Puffer bleiben abgeschaltet, außer wenn ein Lesevorgang geschieht.

1.6.2.6 Periphere-Datenkanäle

Der MCS 6530 verfügt über 16 Anschlüsse für periphere Eingangs/Ausgangs-Vorgänge. Jeder Anschluß kann einzeln durch Software als Eingang oder Ausgang programmiert werden. Die 16 Stifte sind in zwei 8-Bit Kanäle aufgeteilt, PA 0- PA 7 und PBO-PB7. PB5, PB6 und PB7 können auch anderen Zwecken dienen. Das wird in Abschnitt 1.6.4 besprochen. Durch Einschreiben einer 0 in das entsprechende Bit des Datenrichtungs-Registers wirkt der Anschluß als Eingang, bei einer 1 als Ausgang. In der Betriebsart Eingang sind die peripheren Ausgangspuffer im Zustand 1 und die peripheren Datenleitungen werden mit weniger als einer TTL-Last belastet. Bei einem Lesevorgang liest der Prozessor den peripheren Anschluß aus. Das Peripheriegerät empfängt vom MCS 6530 die Information aus dem Datenregister. Alle Peripherie-Anschlüsse sind TTL-kompatibel. Die Anschlüsse PA 0 und PB 0 können bei 1,5 V 3ma aufnehmen. Sie können so einen Darlington-Transistor treiben.

1.6.2.7 Adressleitungen (A 0-A 9)

Es gibt 1o Adressanschlüsse. Zusätzlich gibt es den Anschluß ROM SELECT, der zur Auswahl eines ROM dient. Die genannten Anschlüsse dienen immer zur Adressierung. Weitere zwei Anschlüsse (PB 5 und PB 6) sind Masken programmierbar und können entweder einzeln oder zusammen zur Bausteinauswahl (CHIP SELECTS) verwendet werden. Wenn sie als periphere Datenanschlüsse benutzt werden schließt das eine Verwendung als Bausteinauswahlleitung aus.



**CS1/CS2 ARE MASK OPTIONS IN PLACE OF PB6/PB5*
***PB6 MAY BE USED AT IRQ*

MCS6530 Internal Architecture

ABBILDUNG 1.33: MCS 6530 INNERER AUFBAU

1.6.3. Interne Organisation

Bild 1.33 zeigt ein Blockschaltbild des inneren Aufbaus. Der MCS 6530 ist in vier Abschnitte geteilt: RAM, ROM, Eingang/Ausgang und Zeitgeber. RAM und ROM haben mit dem Prozessor über den Datenbus des Systems und die Adressleitung direkte Verbindung. Der Eingangs/Ausgangs-Abschnitt besteht aus zwei Hälften von 8 Bit. Jede Hälfte enthält ein Datenrichtungs-Register (DDR) und ein Eingangs/Ausgangs-Register. Register DDR steuert die peripheren Ausgangspuffer. Eine in das DDR eingeschriebene 1 setzt den entsprechenden Peripherie-puffer als Ausgang. Dadurch erscheint alles, was im Eingangs/Ausgangs-Register geschrieben wird am entsprechenden Peripherie-Anschluß. Eine in das DDR eingeschriebene 0 hindert den Ausgangspuffer an der Übertragung von Daten vom Eingangs/Ausgangs-Register. Der Ausgangs-puffer bleibt im Zustand H und ist damit bereit Daten von der Peri-pherieleitung zu empfangen.

Es muß bemerkt werden, daß der Prozessor wenn er das Eingangs/Ausgangs-Register ausliest tatsächlich den Peripherie-Anschluß und nicht das Register ausliest. Die Daten im Eingangs/Ausgangs-Register können nur durch einen Schreibvorgang vom Prozessor geändert werden. Das Register wird von den Daten am Peripherieanschluß nicht beeinflußt.

1.6.3.1. ROM - 1 K Byte (8 K Bits)

Das 8 K ROM ist 1024 x 8 organisiert. Zur Adressierung des gesamten ROM sind die Adressleitungen A 0 - A 9 und RS 0 nötig. Unter Zuhilfenahme von CS 1 und CS 2 können bis zu 7 MCS 6530 adressiert werden. Das gibt zusammen 7168 x 8 Bits zusammenhängendes ROM.

1.6.3.2 RAM - 64 Bytes (512 Bits)

Der MCS 6530 enthält ein statisches RAM 64 x 8. Es wird adressiert durch A 0 - A 5 (Byte Select), RS 0, A 6, A 7, A 8, A 9 und abhängig von der Zahl der Bausteine im System CS 1 und CS 2.

1.6.3.3 Interne Peripherie - Register

Es gibt vier interne Register, zwei Datenrichtungs-Register und zwei periphere Eingangs/Ausgangs Datenregister. Die zwei Datenrichtungsregister (Seite A und Seite B) steuern die Richtung der Daten in und aus den peripheren Anschlüssen. Z.B. setzt eine ins Datenregister A Stelle 3 geladene 1 den Peripherie-Anschluß PA 3 als Ausgang. Wenn statt dessen eine 0 geladen worden wäre, würde PA 3 als Eingang arbeiten. Die zwei Dateneingangs/ausgangs-Register werden zum Festhalten der Daten vom Datenbus während eines Schreibvorgangs benutzt, bis das Peripheriegerät die Daten auslesen kann, die der Prozessor geliefert hat. Obwohl der Prozessor während des Lesevorgangs den peripheren Anschluß ausliest ist die Adresse diesselbe wie beim Register. Für die Anschlüsse, die durch das Datenrichtungs-Register als Ausgänge programmiert sind, sind die Daten an den Anschlüssen diesselben wie im Eingangs/Ausgangs-Register.

1.6.3.4 Zeitgeber

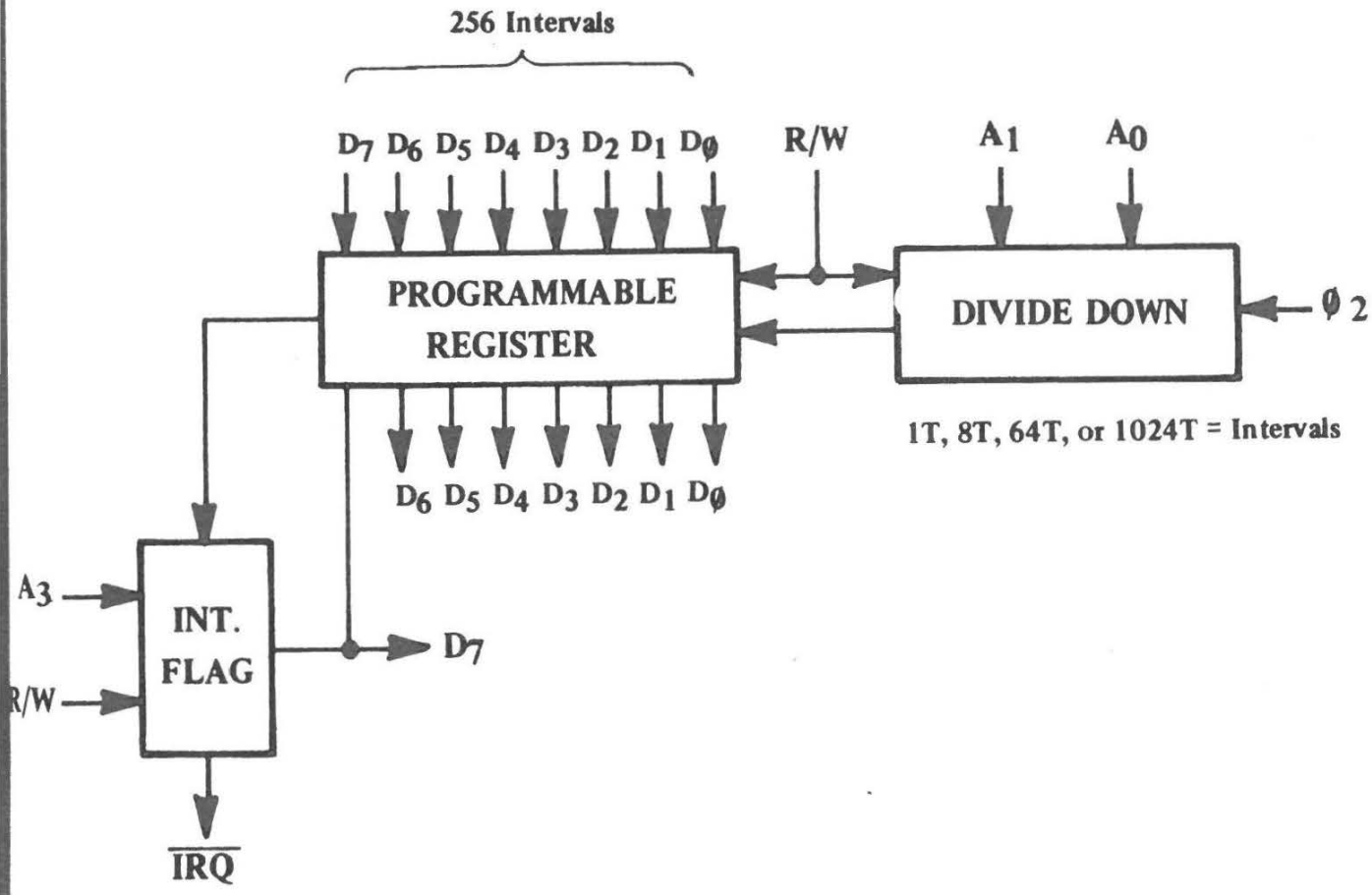
Der Zeitgeber Abschnitt des MCS 6530 enthält 3 Grundbestandteile: ein Register zum vorläufigen Herunterteilen, ein programmierbares 8-Bit Register und eine Interrupt Logik. Sie sind in Bild 1.34 dargestellt.

Der Zeitgeber kann zur Zählung von bis zu 256 Zeitabschnitten programmiert werden. Jeder Zeitabschnitt kann entweder 1, 8, 64 oder 124 Systemtakte lang sein. Bei Zählerüberlauf wird ein Interrupt-Bit auf logisch Eins gesetzt. Darauf wird mit dem internen Takt bis maximal - 255 abwärts gezählt. Durch einen Lesevorgang kann festgestellt werden, wieviel Takte (max. 255) seit dem Setzen vergangen sind.

Beim Einschreiben in den Zeitgeber werden die höherwertigen 8 Bit des Zeitgebers vom Systemdatenbus geschrieben. Wenn 52 Zeitabschnitte gezählt werden sollen, muß in den Zeitgeberabschnitt 0 0 1 1 0 1 0 0 eingeschrieben werden. Zu dieser Zeit werden die Zeitabschnitte 1, 8, 64 oder 124 Taktperioden von den Adressleitungen A 0 und A 1 dekodiert. Wenn Adressleitung A 3 während des Schreibens auf H liegt, wird das Interrupt-Bit auf Anschluß PB 7 geschaltet. PB 7 sollte als Eingang bei Verwendung als Interrupt-Anschluß programmiert sein. Er geht H bei Interrupt. Wird der Zeitgeber vor dem Setzen des Interrupt-Bit abgefragt, wird die Zahl der verbleibenden Zeitabschnitte (Z.B. 51, 50, 49 ...) ausgelesen.

Wenn der Zeitgeber zur Zeit des Interrupt ausgelsen wird, ergeben sich 1 1 1 1 1 1 1 1. Nach dem Interrupt zählt das Register des Zeitgebers mit dem Systemtakt abwärts. Wenn der Zeitgeber nach einem Interrupt gelesen wird und der Wert 1 1 1 0 0 1 0 0 erscheint, ist die Zeit seit dem Interrupt 28 Taktperioden der ausgegebene Wert ist im Zweier-komplement.

Ausgelesener Wert	=	1 1 1 0 0 1 0 0	
Komplement	=	0 0 0 1 1 0 1 1	
ADD 1	=	0 0 0 1 1 1 0 0	= 28.



Basic Elements of Interval Timer

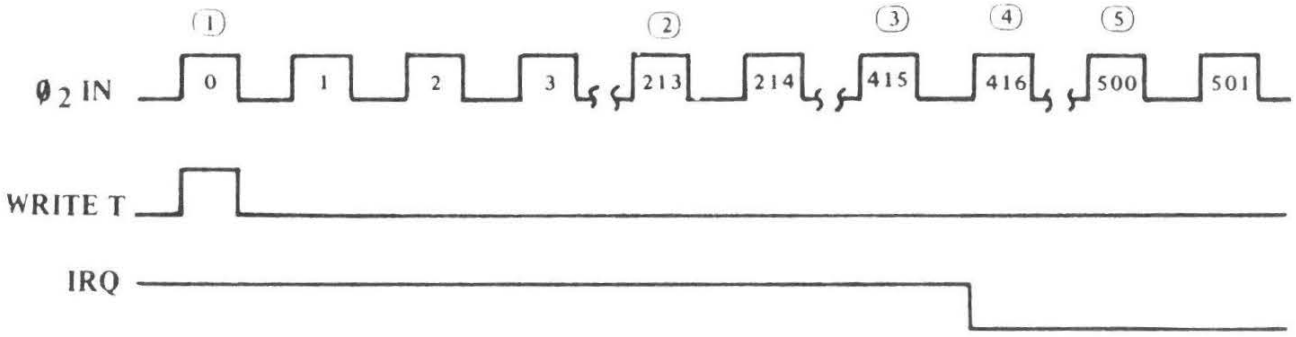
ABBILDUNG 1.34: GRUNDBESTANDTEILE DES ZEITGEBERS

Um die gesamte abgelaufene Zeit zu erhalten braucht nur das Zweierkomplement der ursprünglich in den Zeitgeber eingeschriebenen Zeit addiert zu werden. Nehmen wir an die eingeschriebene Zeit sei 0 0 1 1 0 1 0 0 (= 52). Mit einer Teilung durch 8 ist die Gesamtzeit zum Interrupt (52 x 8) plus 1 = 417. Die gesamte abgelaufene Zeit wäre dann 416 + 28 = 444 Taktperioden, wenn wir annehmen daß der Wert nach dem Interrupt als 1 1 1 0 0 1 0 0 ausgelesen wurde.

Nach dem Interrupt wird er zurückgesetzt, immer wenn in den Zeitgeber eingeschrieben oder aus ihm ausgelesen wird. Die Interrupt Flag wird jedoch nicht zurückgesetzt, wenn Lesen oder Schreiben des Zeitgebers zur gleichen Zeit mit dem Interrupt geschehen. Bild 1.35 zeigt ein Beispiel eines Interrupts. Wenn der Zeitgeber nach einem Interrupt ausgelesen wird, muß A 3 auf L liegen, um den Anschluß $\overline{IRQ}$ abzuschalten. Das wird zur Vermeidung weiterer Interrupts bis zu einem neuen Schreibvorgang in den Zeitgeber getan.

1.6.4 Adressierung

Zur größeren Flexibilität bietet die Adressierung des MCS 6530 viele verschiedene Möglichkeiten. Der Benutzer kann sein System mit RAM im niederen Speicherbereich, ROM im höheren Speicherbereich und Eingangs/Ausgangs-Registern mit Zeitgebern zwischen den Grenzwerten zusammenstellen. Es gibt 10 Adressleitungen (A 0 - A 9). Drei weitere Adressleitungen können zur Bausteinwauswahl und zur Unterscheidung zwischen ROM, RAM, Eingang/Ausgang und Zeitgeber benutzt werden. Zwei der zusätzlichen Leitungen sind die Bausteinauswahl-Leitungen 1 und 2 (CS 1 und CS 2). Ebenso können PB 5 und PB 6 Bausteinauswahl-Anschlüsse sein. Ob die Anschlüsse zur Bausteinauswahl oder als periphere Eingangs/Ausgangs Anschlüsse benutzt werden hängt von der Maske ab und muß bei der Bestellung angegeben werden. Die beiden Anschlüsse wirken unabhängig voneinander. Jeder von ihnen oder beide können als Bausteinauswahl bestimmt werden. Die dritte zusätzliche Adressleitung ist RS 0.



SHOULD THE PROGRAMMABLE TIMER REGISTER BE READ AT THE TIMES NOTED ON THE DIAGRAM ABOVE, IT WOULD CONTAIN:

① Data written into interval timer is $001100100 = 52_{10}$ A divide by 8 pre-scale is used.

② $00011001 = 25_{10}$ $52 - \frac{213}{8} - 1 = 52 - 26 - 1 = 25$

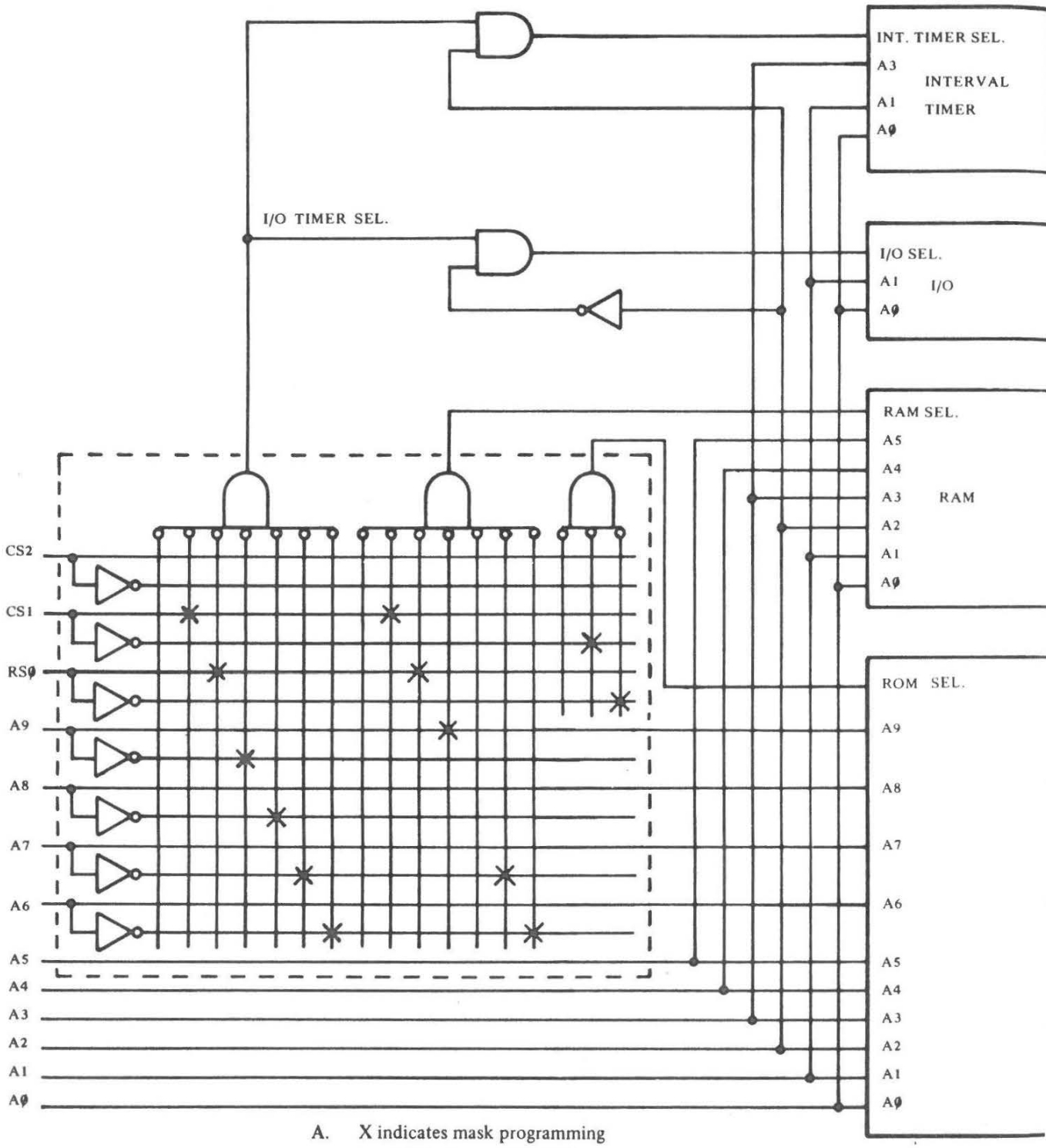
③ $00000000 = 0_{10}$ $52 - \frac{415}{8} - 1 = 52 - 51 - 1 = 0$

④ Interrupt has occurred at Ø2 pulse #416

⑤ 10101100 Two; complement = $01010100 = 84_{10}$ $84 + (52 \times 8) = 500_{10}$

Example of Interrupt Generated by Interval Timer

ABBILDUNG 1.35: BEISPIEL EINES INTERRUPT, AUSGELÖST VOM ZEITGEBER



A. X indicates mask programming
i.e. ROM select = $\underline{CS1} \bullet RSO$
RAM select = $\overline{CS1} \bullet \overline{RSO} \bullet \overline{A9} \bullet A7 \bullet A6$
I/O TIMER SELECT = $\overline{CS1} \bullet \overline{RSO} \bullet A9 \bullet A8 \bullet A7 \bullet A6$

B. Notice that A8 is a don't care for
RAM select

C. CS2 can be used as PB5 in this example.

MCS6530 One Chip Address Encoding Diagram

ABBILDUNG 1.36: MCS 6530 KODIER-DIAGRAMM FÜR EINEN BAUSTEIN

The addressing of the ROM select, RAM select and I/O Timer select lines would be as follows:

		CS2 A12	CS1 A11	RSØ A10	A9	A8	A7	A6
MCS6530 #1,	ROM SELECT	0	0	1	X	X	X	X
	RAM SELECT	0	0	0	0	0	0	0
	I/O TIMER	0	0	0	1	0	0	0
MCS6530 #2,	ROM SELECT	0	1	0	X	X	X	X
	RAM SELECT	0	0	0	0	0	0	1
	I/O TIMER	0	0	0	1	0	0	1
MCS6530 #3,	ROM SELECT	0	1	1	X	X	X	X
	RAM SELECT	0	0	0	0	0	1	0
	I/O TIMER	0	0	0	1	0	1	0
MCS6530 #4,	ROM SELECT	1	0	0	X	X	X	X
	RAM SELECT	0	0	0	0	0	1	1
	I/O TIMER	0	0	0	1	0	1	1
MCS6530 #5,	ROM SELECT	1	0	1	X	X	X	X
	RAM SELECT	0	0	0	0	1	0	0
	I/O TIMER	0	0	0	1	1	0	0
MCS6530 #6,	ROM SELECT	1	1	0	X	X	X	X
	RAM SELECT	0	0	0	0	1	0	1
	I/O TIMER	0	0	0	1	1	0	1
MCS6530 #7,	ROM SELECT	1	1	1	X	X	X	X
	RAM SELECT	0	0	0	0	1	1	0
	I/O TIMER	0	0	0	1	1	1	0

* RAM select for MCS6530 #5 would read = $\overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot A8 \cdot \overline{A7} \cdot \overline{A6}$

MCS6530 Seven Chip Addressing Scheme

ABBILDUNG 1.37: MCS 6530 ADRESSEN-SCHEMA FÜR SIEBEN BAUSTEINE

ADDRESSING DECODE

	ROM SELECT	RAM SELECT	I/O TIMER SELECT	R/W	A3	A2	A1	AØ
READ ROM	1	0	0	1	X	X	X	X
WRITE RAM	0	1	0	0	X	X	X	X
READ RAM	0	1	0	1	X	X	X	X
WRITE DDRA	0	0	1	0	X	0	0	1
READ DDRA	0	0	1	1	X	0	0	1
WRITE DDRB	0	0	1	0	X	0	1	1
READ DDRB	0	0	1	1	X	0	1	1
WRITE PER. REG. A	0	0	1	0	X	0	0	0
READ PER. REG. A	0	0	1	1	X	0	0	0
WRITE PER. REG. B	0	0	1	0	X	0	1	0
READ PER. REG. B	0	0	1	1	X	0	1	0
WRITE TIMER								
÷ 1T W/IRQ to PB7	0	0	1	0	1	1	0	0
÷ 8T WO/IRQ to PB7	0	0	1	0	0	1	0	1
÷ 64T W/IRQ to PB7	0	0	1	0	1	1	1	0
÷ 1024T WO/IRQ to PB7	0	0	1	0	0	1	1	1
READ TIMER DISABLE IRQ TO PB7	0	0	1	1	0	1	X	0
READ INTERRUPT FLAG	0	0	1	1	X	1	X	1

Addressing Decode for I/O Register and Timer

ABBILDUNG 1.38: ADRESSIER-DEKODIERUNG FÜR E/A-REGISTER UND ZEITGEBER

In einem 2-chip System würden der MCS 6502 und der MCS 6530 RS 0 benutzen, um zwischen den Abschnitten des MCS 6530 zu unterscheiden die ROM und nicht ROM sind. Mit den verfügbaren Adressanschlüssen können 7 K zusammenhängendes ROM ohne externe Dekodierung adressiert werden. Unten wird je ein Beispiel für ein Adressierverfahren mit einem Baustein und mit 7 Bausteinen MCS 6530 angegeben.

1.6.4.1 Adressierung für einen Baustein

Bild 1.36 veranschaulicht die Systemdekodierung für einen Baustein MCS 6530.

1.6.4.2. Adressierung für sieben Bausteine

Im System mit sieben Bausteinen ist das Ziel 7 K zusammenhängendes ROM zu haben und das RAM im unteren Speicherbereich. Die 7 K ROM sollten zwischen den Adressen 65.536 und 1024 liegen. In diesem Fall nehmen wir an, daß A 13, A 14 und A 15 alle 1 sind, wenn ROM adressiert wird und 0 wenn RAM oder Eingang/Ausgang adressiert wird. Damit liegt das 7 K ROM zwischen den Adressen 65.535 und 57.367. Die zwei als Bausteinauswahl oder Eingang/Ausgang verfügbaren Anschlüsse würden als Bausteinauswahl-Anschlüsse Masken programmiert werden. Anschluß RS 0 würde mit der Adreßleitung A 10 verbunden werden. Anschlüsse CS 1 und CS 2 würden jeweils mit den Adreßleitungen A 11 und A 12 verbunden werden. Siehe Bild 1.37.

Die beiden gezeigten Beispiele erlauben die Adressierung von ROM und RAM. Wenn jedoch der Eingangs/Ausgangs-Zeitgeber adressiert worden ist, ist weitere Dekodierung notwendig, um das gewünschte Eingangs/Ausgangs-Register auszuwählen ebenso die Kodierung des Zeitgebers.

1.6.4.3 Adressierung des Eingangs/Ausgangs-Registers und des Zeitgebers

Bild 1.38 zeigt die Dekodierung zum adressieren der internen Bestandteile und die Programmierung des Zeitgebers. Adressleitung A 2 unterscheidet die Eingangs/Ausgang-Register vom Zeitgeber. Wenn A 2 auf L liegt und die Anwahl des Eingangs/Ausgangs Zeitgebers auf H, werden die Eingangs/Ausgangs-Register adressiert. Wenn einmal die Eingangs/Ausgangs-Register adressiert sind, dekodieren die Adressleitungen A 1 und A 0 das gewünschte Register.

Wenn der Zeitgeber ausgewählt ist dekodieren A 1 und A 0 die Matrix für den Teilfaktor. Dies wird in dem Abschnitt über den Zeitgeber weiter besprochen. Außerdem wird die Adresse A 3 benutzt die Interrupt Flag an PB 7 einzuschalten.

KAPITEL 2

DIE KONFIGURATION DES MIKROCOMPUTER-SYSTEMS

2.1 DIE AUFGABE DER SYSTEM-KONFIGURATION

Der erste Teil jeder Entwicklung mit einem Mikroprozessor ist die Festlegung der Konfiguration. Das erfordert wahrscheinlich mehr Kreativität vom Entwickler als jeder andere Teil der Entwicklung. Das Ziel ist die Aufstellung einer Liste der Bauteile, aus denen das System besteht, ein genauer Plan der Verbindungen und eine genaue Beschreibung der Gesamtarbeitsweise des Systems. Das schließt eine Definition ein, wie der Prozessor die Peripherie-Bausteine steuert und welche internen Operationen ausgeführt werden. Eine genaue Ausführung des Entwurfs, wie die Auslegung gedruckter Leiterplatten und das Schreiben von Programmen ist noch nicht nötig. Die gesamte Arbeitsweise sollte aber genau genug untersucht werden, um sicherzustellen, daß das System nach Herstellung von Hard- und Software richtig arbeitet.

Zur technischen Auswahl der Bauteile und der Festlegung der Gesamtarbeitsweise des Systems müssen zwei Faktoren berücksichtigt werden:

1. System-Geschwindigkeit
2. System-Eingang/Ausgang

Beide Faktoren beeinflussen sich gegenseitig. Es ist daher normalerweise nötig, eine Eingangs/Ausgangs-Konfiguration festzulegen und dann zu überprüfen, ob der Prozessor mit der nötigen Geschwindigkeit für die Peripherie-Bausteine arbeiten kann. Wenn mit der Eingangs/Ausgangs-Arbeitsweise Schwierigkeiten auftreten, muß diese Struktur neu festgelegt und untersucht werden.

Außer den durch die Eingangs/Ausgangs-Bausteine gestellten Geschwindigkeitsanforderungen gibt es allgemeine Anforderungen an die Geschwindigkeit für die internen Arbeitsvorgänge des Prozessors (arithmetische Operationen, Handhabung von Daten usw). Diese Anforderungen sind üblicherweise nicht so streng, wie die mit Eingang/ Ausgang verbundenen. Sie sollten aber zusammen mit den anderen Erfordernissen des Systems festgelegt werden. Die abschließende Überprüfung der Systemgeschwindigkeit kann erst nach Herstellung von Hardware und Programm stattfinden. Um zu verhindern, daß während der letzten Stufen der Entwicklung Probleme entstehen, müssen die Erfordernisse und Möglichkeiten des Systems sehr früh in der Entwicklung untersucht werden.

2.2 EINGANGS/AUSGANGS-VERFAHREN

2.2.1 Der Eingangs/Ausgangs-Kanal für allgemeine Anwendung

Das Konzept des E/A-Kanals wurde in Abschnitt 1 kurz eingeführt und die Arbeitsweise der zwei Bausteine aus der Familie MCS 6500, die E/A-Möglichkeiten für allgemeine Anwendung bieten, sind in Abschnitt 1.5 und 1.6 besprochen worden. Es ist jedoch wenig darüber gesagt worden, welche Faktoren zur Konfiguration einer E/A-Struktur bei Benutzung dieser Bausteine beachtet werden mussen.

Der E/A-Kanal für allgemeine Anwendung besteht aus acht Leitungen, von denen jede entweder als Eingang oder als Ausgang arbeiten kann. Als Eingang kann jeder Leitung die Stellung eines Schalters oder 1 Bit Daten erkennen. Als Ausgang kann sie eine Lampe, eine Arbeitsspule usw. steuern oder 1 Bit Daten an ein Peripherie-Gerät geben. Wenn dieses Verfahren zur Steuerung der Peripherie benutzt wird ist die Wirkungsweise jeder Leitung vollständig im System-Programm festgelegt.

Für die meisten Systeme bietet der Mehrzweck-Schnittstellen-Baustein eine mehr als ausreichende Geschwindigkeit und Flexibilität zur Lösung des gesamten Peripherie-Schnittstellen-Problems. Meistens können durch die kleineren Bauteilkosten und die vereinfachte Lagerhaltung mit nur einem Typ Schnittstellen-Baustein Kosten eingespart werden. Außerdem kann der Entwickler mit einem Mehrzweck-Peripherie-Schnittstellen-Baustein die Arbeitsweise des Schnittstellen-Bausteins genau auf das vorliegende Problem zuschneiden.

Der abschließenden Bauteilauswahl geht eine Untersuchung jedes Abschnitts der E/A-Struktur des Systems und der Leistung des Gesamtsystems voraus. Schließlich wird unter Berücksichtigung von Kosten und Leistung des Systems eine Zusammenstellung von Mehrzweck- und Spezial-Schnittstellen-Bausteinen ausgewählt.

Die Arbeitsgeschwindigkeit des Prozessors hängt von zwei Dingen ab. Das Erste ist einfach die Zahl der Befehle zur Ausführung der gewünschten Arbeitsgänge. Das Zweite ist der Zeitanteil, der zur Bedienung von Interrupts gebraucht wird. Ein typisches System verwendet z.B. verschiedene Interrupt-Signale, die in festen Zeitabständen auftreten. Diese können manchmal mit anderen Interrupts von einem Peripherie-Baustein zusammengefaßt werden. Die gesamte Zeit zur Bedienung dieser Interrupts darf nur so groß werden, daß der Prozessor noch ausreichend Zeit zur Ausführung des Haupt-Programms mit der erforderlichen Geschwindigkeit hat.

Während der System-Konfiguration müssen keine ins einzelne gehende Programme geschrieben werden. Es wird aber nötig sein kleine Abschnitte der Software zur Überprüfung der Geschwindigkeit der Ausführung und zur Sicherstellung der richtigen Arbeitsweise des gesamten Systems zu schreiben.

In diesem Kapitel werden die Verfahren zur Steuerung der verschiedenen Bauteile in einem Mikrocomputer-System und der Peripherie-Bausteine, die mit ihm verbunden sind, besprochen. Eine Besprechung von Programmier-Verfahren zur Optimierung der Leistung des Gesamtsystems ist in der Programmier-Fibel enthalten.

2.2.2 Der Peripherie-Schnittstellen-Baustein für besondere Anwendungen

In jedem Mikrocomputer-Entwurf muß der für spezielle Anwendungen gedachte E/A-Baustein als Möglichkeit beachtet werden. Diese Bausteine sind zur Lösung eines einzigen, genau definierten Problem entwickelt worden. Sie können z.B. einen bestimmten Drucker

ansteuern, eine bestimmte Nachrichten-Leitung betreiben oder eine abgetastete Anzeige treiben. Diese Spezial-Bausteine können ihre spezielle Aufgabe mit sehr wenig Unterstützung durch den Prozessor erfüllen. Der Hauptvorteil dieser Art Schnittstellen-Baustein ist die geringe Belastung des Prozessors. Der große Nachteil sind die größeren Bauteilkosten. Die Produktionsstückzahl für diese Bauteile ist kleiner als für die allgemein verwendbaren E/A-Bausteine. Im allgemeinen ist auch die Chip-Fläche größer.

Die Anwendung spezieller Schnittstellen-Bausteine wird in diesem Handbuch nicht besprochen. Stattdessen wird das allgemeine Problem der Konfiguration eines 8 Bit Peripherie-Kanals für zwei Richtungen untersucht. Außerdem erördert dieses Kapitel einige besondere Verfahren, die die Leistung dieser Art Schnittstellen-Baustein vergrößern.

2.2.3 Konfiguration des Mehrzweck-E/A-Kanals

Jede Leitung im 8 Bit Peripherie-Steuerungs-Kanal im MCS 6520 und MCS 6530 kann als Eingang oder Ausgang programmiert werden. Der Prozessor schreibt dazu ein Muster von Einsen und Nullen in das Datenrichtungs-Register. Einschreiben einer 1 programmiert einen Ausgang, eine 0 einen Eingang. Dieser Arbeitsgang wird normalerweise nur während der Initialisierung des Systems ausgeführt. Er kann aber auch unter Programmsteuerung erfolgen und erlaubt so einige wichtige Verfahren zu Peripheriesteuerung. Dafür wird unten ein Beispiel geschrieben.

Zur Konfiguration des Mehrzweck-E/A-Kanals werden zunächst die Peripherie-Geräte untersucht, um die verschiedenen Steuereingänge, Schalter, Fühler, Datensignale usw. festzustellen, die der Prozessor zur Steuerung des Gerätes verarbeiten muß. Dann wird jeder Funktion eine Leitung im E/A-Kanal zugewiesen. In einer Liste werden Funktion, Eingang oder Ausgang, festgelegt.

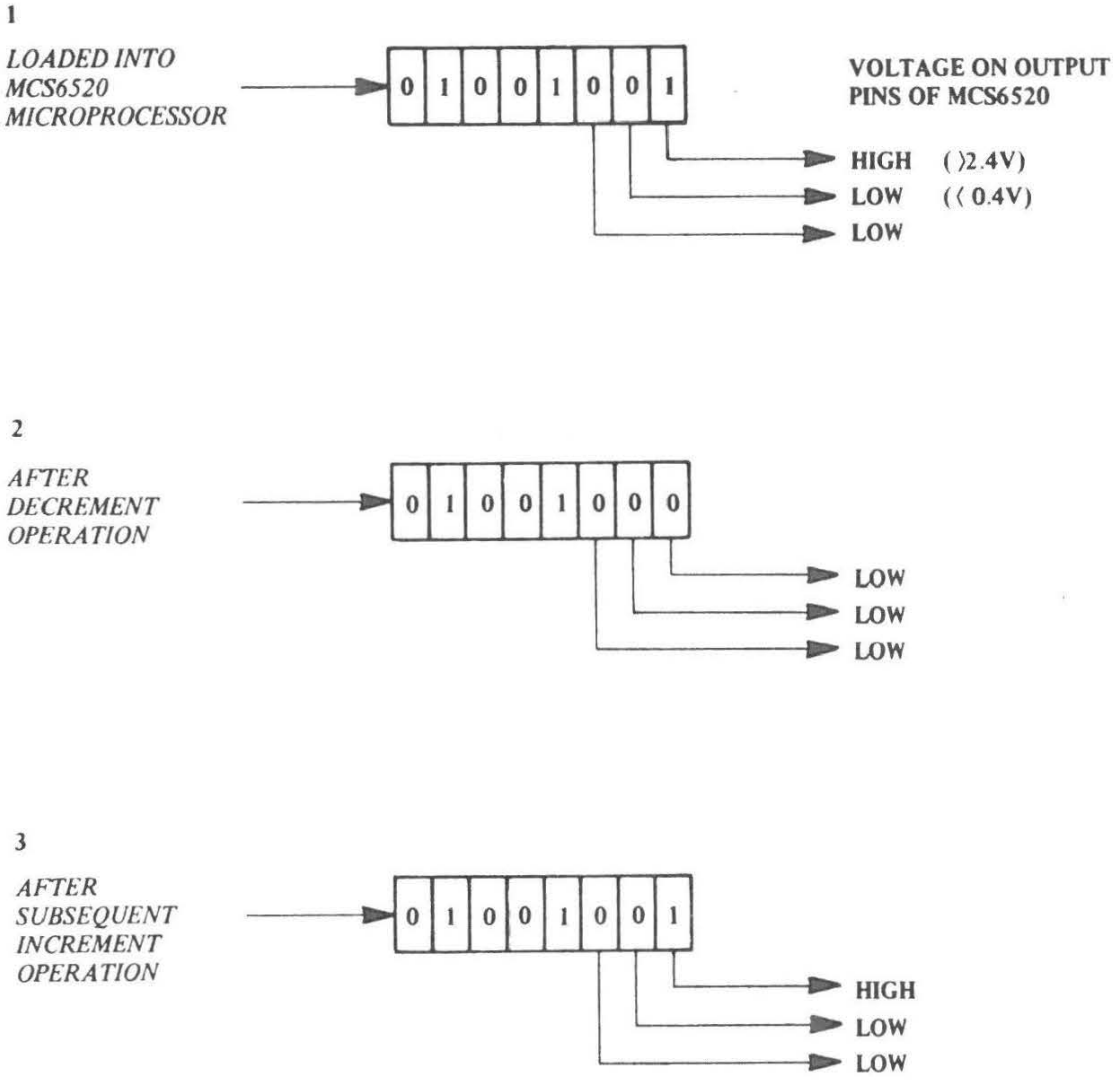
Die tatsächliche Zuordnung kann sehr spät in der Entwicklung geschehen, nachdem Softwareverfahren und die Auslegung gedruckter Leiterplatten erwogen worden sind, da sich die Anschlüsse sehr wenig voneinander unterscheiden. Tatsächlich können Software-Überlegungen der einzige Grund sein, ein Signal mit einem bestimmten Anschluß zu verbinden.

Zum tiefergehenden Verständnis der Software im System MCS 6500 muß die Programmier-Fibel genau durchgearbeitet werden. Einige Arbeitsgänge des Prozessors, die die Zuordnung von Ein- und Ausgängen beeinflussen, werden hier kurz besprochen.

2.2.3.1 Zuordnung von Ausgängen

Ein wichtiger Punkt bei der Zuordnung von Ausgangs-Anschlüssen kann die Fähigkeit des Prozessors MCS 650X sein, die Speicheradresse aufwärts und abwärts zu zählen (increment und decrement). Da der E/O-Kanal als Speicherstelle behandelt wird, kann dieses auf- und abzählen der Adresse zum schnellen Setzen und Löschen des niederwertigen Bits an dieser Speicherstelle dienen, wie in Bild 2.1. gezeigt.

Dies Verfahren beeinflußt nichts als das unterste Bit, wenn es richtig angewandt wird. Diese Operation kann schneller ausgeführt werden als verschiedene andere Software-Verfahren, die ein einzelnes Bit beeinflussen. Die Steuerung eines einzelnen Anzeigers, einer Daten-Leitung usw. kann sehr verbessert werden, wenn man sie auf das unterste Bit eines E/A-Kanals legt. Deswegen kann das unterste Bit der beiden Peripherie-Kanäle (PA0 und PB0) im MCS 6530 direkt Transistoren treiben. Ein einfacher Transistor an einem dieser Anschlüsse kann in vielen Anwendungen eine sehr zweckmäßige Steuerung für einen Motor, eine Lampe usw. ergeben.



Control of Low Order Bit of MCS6520 Output Register

ABBILDUNG 2.1: STEUERUNG DES UNTERGEORDNETEN BITS IM MCS 6520 AUSGABE-REGISTER

Die Fähigkeit des Prozessors, Daten im Speicher zu verschieben, ist ein weiterer wichtiger Punkt bei der Zuordnung von Ausgängen. Vorgänge, die aufeinanderfolgende Strobe-Signale brauchen, können durch verschieben eines einzigen Signals auf H (oder L) von Stift zu Stift durch die Software gesteuert werden. Die Auswahl der Anschlüsse kann die Steuerung dieses Signals stark vereinfachen.

2.2.3.2 Zuordnung von Eingängen

Im allgemeinen liest der Prozessor die Eingangsdaten von Schaltern, Tastaturen usw. über den E/A-Kanal in die internen Register (meistens den Akkumulator) ein. Das ausgeführte Programm wird vom daraus ergebenden Zustand der Bits im Prozessor-Status-Register gesteuert. Während dieser Übertragung word das Bit N im Prozessor-Status-Register gleich dem obersten Bit (Bit 7) des ausgelesenen Wortes vom E/O-Kanal gesetzt. Dies Bit N kann dann verwendet werden, um den Prozessor verschiedene Abschnitte des Programms ausführen zu lassen (Siehe Programmier-Fibel, Kapitel 4 Branching). Ebenso kann durch Ausführung bestimmter Befehle das Bit V im Prozessor-Status-Register dem Bit 6 im E/A-Kanal gleichgesetzt werden. Dieses Bit kann zur Beeinflussung des ausgeführten Programms benutzt werden.

Das Setzen der internen Bits von Bit 6 und 7 eines Speicherwortes erlaubt eine einfache Überprüfung des Zustandes von Schaltern, Fühlern usw. die an diese zwei Leitungen eines E/A-Kanals angeschlossen sind, wenn diese als Eingänge programmiert sind. Wenn mehr als zwei Eingangs-Signale mit dem Kanal verbunden werden sollen, legt man sie auf Bit 5, dann Bit 4 usw. Der Prozessor kann dann Operationen ausführen, die die niederwertigen Bits eines nach dem anderen in Bit 7 schieben und das Bit N diesem Bit gleichsetzen.

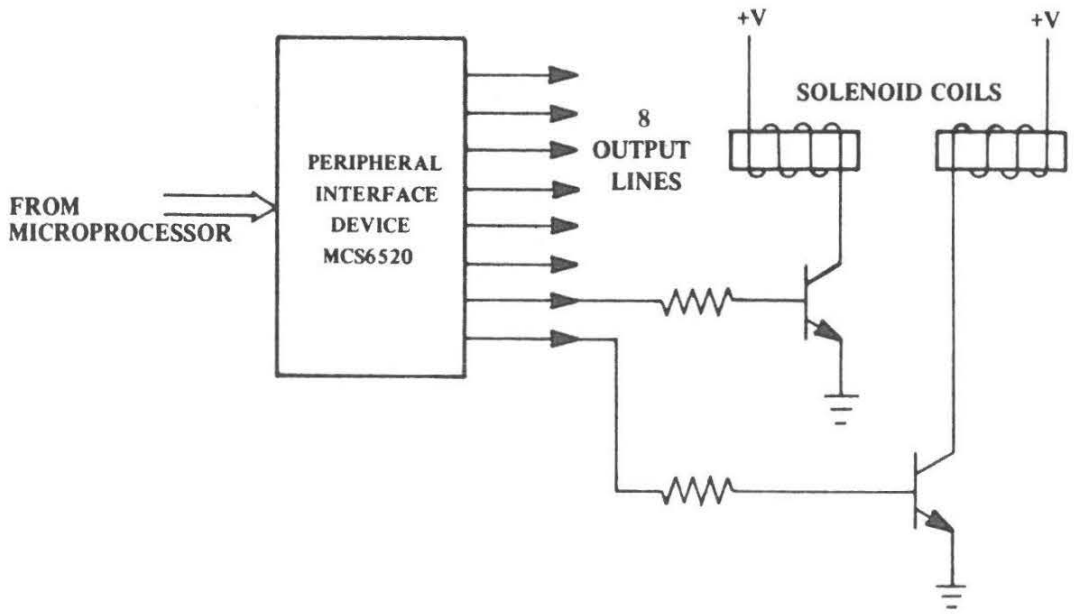
Nach jeder Verschiebung kann das Bit N benutzt werden, um das tatsächlich auszuführende Programm zu bestimmen. (Die Verschiebe-Befehle (Shift) werden in der Programmier-Fibel besprochen).

Am obigen Beispiel sieht man, daß die Zuordnung stark von dem Software-Verfahren abhängt, die zur Steuerung jedes Anschlusses verwendet werden. Der Entwickler muß mit diesem Verfahren vertraut sein und das geplante Verfahren aufzeichnen, wenn er die Zuordnung festlegt. Das ist besonders wichtig, wenn das Systemprogramm von einer anderen Person geschrieben werden soll. Der Systementwickler muß die Eingangs/Ausgangs-Struktur ständig überprüfen, um die Software zu optimieren während das Systemprogramm geschrieben wird.

2.2.4 Einschalten der Stromversorgung

In Kapitel 1, Abschnitt 1.3.3 wird die Wirkungsweise der Funktion RESET auf das System besprochen. Sie kann benutzt werden, damit alle E/A-Leitungen beim Einschalten in einem bekannten Zustand sind. Der Entwickler muß sichergehen, daß dieser Zustand RESET die Peripherie-Geräte nicht nachteilig beeinflußt. Dieser Abschnitt bespricht einige Probleme die beim Rücksetzen des Systems austreten können und bespricht verschiedene Verfahren, die für einen reibungslosen Einschaltvorgang verwendet werden können.

Die E/A-Leitungen des MCS 6530 und MCS 6520 gehen alle in den Eingangs-Zustand, wenn die Rücksetz-Leitung auf L geht. Die Anschlüsse der E/A-Leitungen des MCS 6530 und die Anschlüsse des Peripherie-Kanals A beim MCS 6520 gehen auf Vdd. Das liegt an der Ausgangs-Struktur dieser Anschlüsse. Im Eingangs-Zustand sind die Ausgangsschalter offen, aber das Last-Bauelement liefert einen Strom in den Anschluß.



MCS6520 Control of Transistor Driven Solenoids

ABBILDUNG 2.2: MCS 6520 STEUERUNG DER MIT TRANSISTOR GESCHALTETEN MAGNETSPULEN

MCS6520 Control of PNP Transistor Driving Solenoid Coil

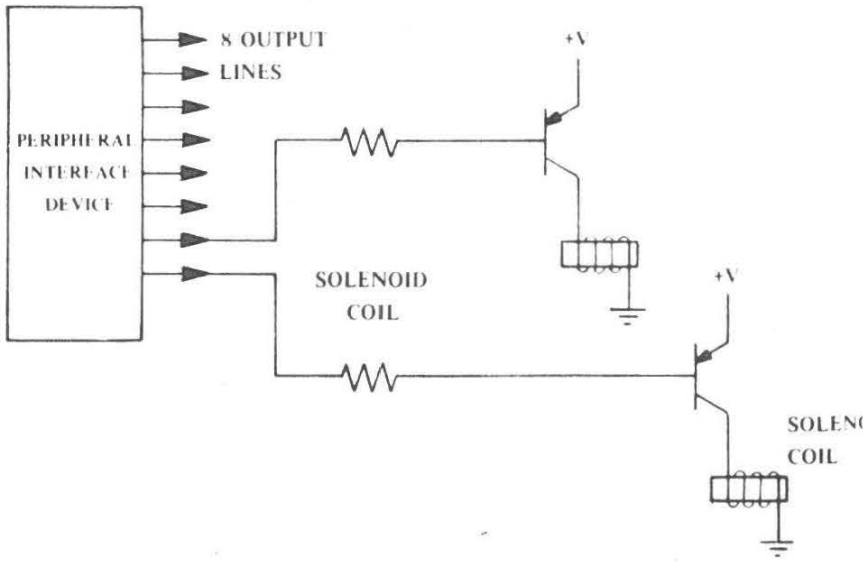


ABBILDUNG 2.3A: MCS6520 STEUERUNG EINES PNP-TREIBER-TRANSISTORS FÜR EINE MAGNETSPULE

MCS6520 Controlling Both Power and Drivers of Solenoid Cell

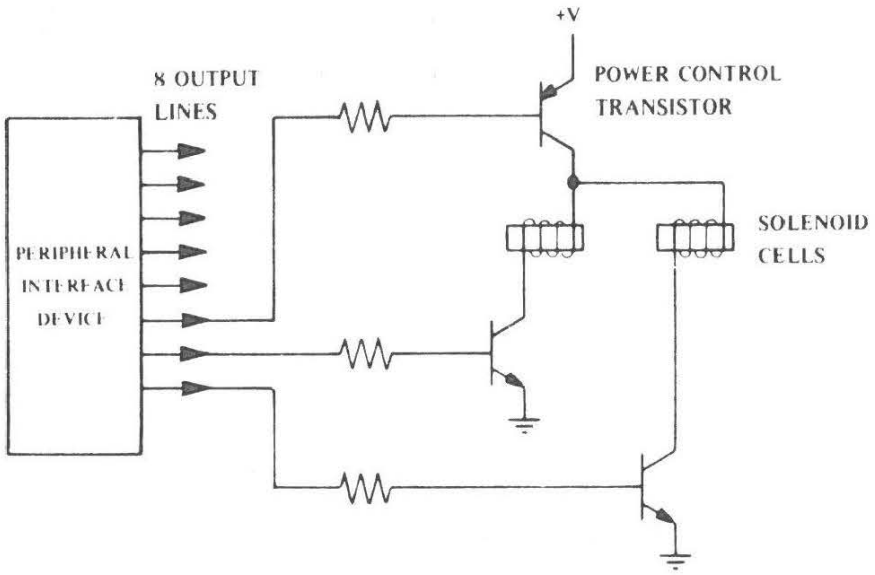


ABBILDUNG 2.3B: MCS 6520 STEUERUNG DER VERSORGUNGSSPANNUNG UND DER MAGNETSPULE

MCS6520 Driving TTL Buffers

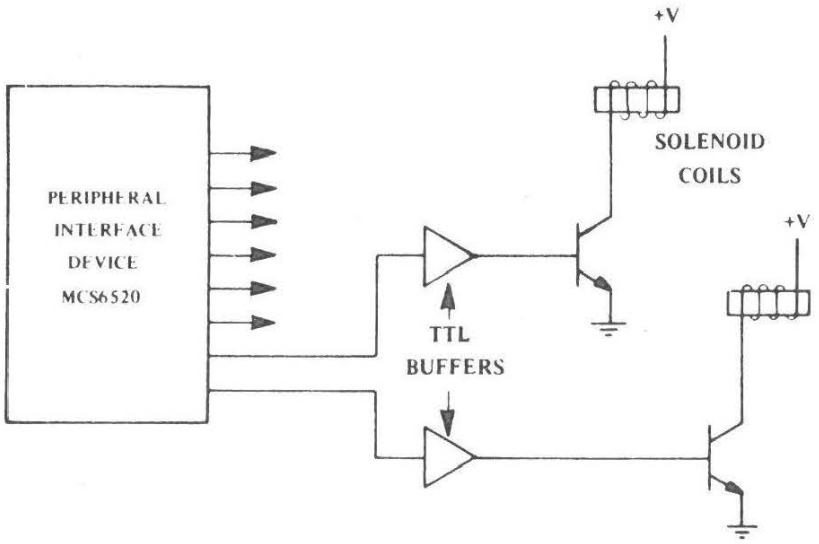
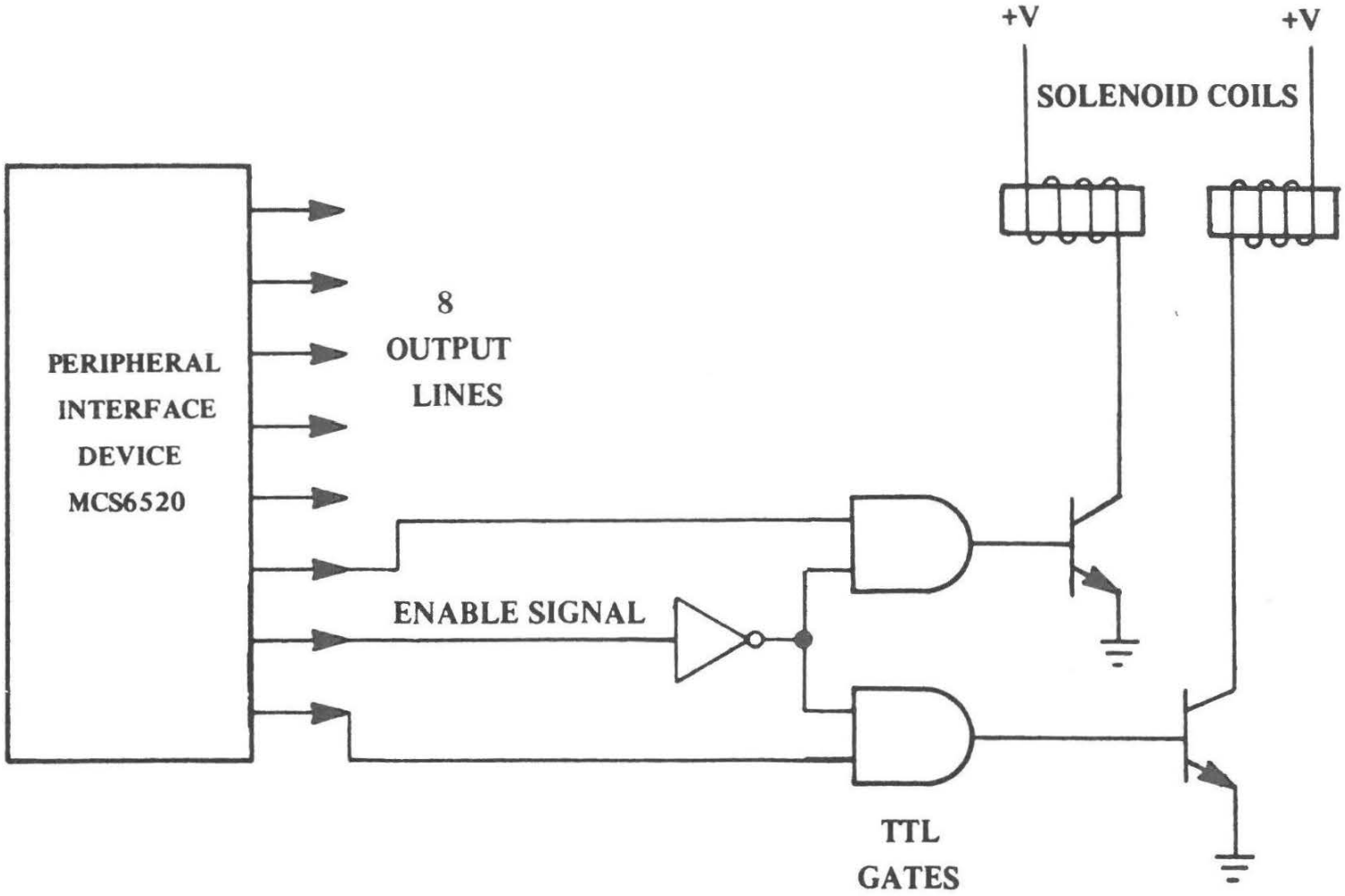


ABBILDUNG 2.4: MCS 6520 TREIBEN DERT TTL-PUFFER



MCS6520 Controlling Solenoids with Enable Signal and TTL Interface

ABBILDUNG 2.5: MCS 6520 STEUERUNG DER MAGNETSPULEN MIT ENABLE-SIGNALEN UND TTL-INTERFACE

Bild 2.2. zeigt einen Peripherie-Kanal, der zwei Arbeitsspulen treibt. Diese Arbeitsspulen können nach Initialisierung des Systems richtig gesteuert werden. Wenn jedoch der Schalter für die Rücksetzung von Hand betätigt wird, gehen die Anschlüsse in den Eingangs-Zustand, die Transistoren schalten und die Magnetspulen werden betätigt. Das kann in den meisten mechanischen Subsystemen verhängnisvoll sein. Es ist daher wichtig, daß dieser mögliche Zustand verstanden und verhindert wird. Bild 2.3 zeigt zwei zufriedenstellende Lösungen für dieses Problem. Bei der Ersten, Bild 2.3a, muß der Prozessor eine 0 in die Ausgangsleitung schreiben, um die Arbeitsspulen zu betätigen. Damit ist sichergestellt, daß die Arbeitsspulen nicht gleichzeitig eingeschaltet werden, wenn der Schalter für die Rücksetzung von Hand gedrückt wird. Es tritt jedoch ein anderes mögliches Problem auf. Wenn die Rücksetz-Leitung des peripheren Schnittstellen-Bausteins auf L geht, wird der Inhalt des Peripherie-Daten-Registers und des Datenrichtungs-Registers gelöscht. Wenn im Datenrichtungs-Register Einsen stehen, werden beide Arbeitsspulen sofort betätigt, da im Peripherie-Daten-Register eine Null steht. Wenn die System-Software zunächst die Bits im Peripherie-Daten-Register auf 1 und dann das Datenrichtungs-Register auf 1 setzt, kann das vermieden werden. Der E/A-Anschluß geht auf H, wenn der Rücksetz-Schalter betätigt wird und bleibt einfach während der Routine zur Initialisierung auf H.

Bild 2.3b zeigt eine Lösung, die bei einem großen System oder einer komplizierten Peripherie besser anwendbar ist. Hier wird eine getrennte Ausgangsleitung zum Anlegen der Stromversorgung an das Peripherie-Gerät benutzt. Die Stromversorgung für das gesamte Peripherie-Gerät oder nur die kritischen Bauteile bleibt abgeschaltet, bis das gesamte System initialisiert ist und das Systemprogramm laufen kann. Im Eingangszustand laufen die E/A-Leitungen des Peripherie-Kanals B des MCS 6520 leer (hohe Impedanz).

Es treten daher in der Konfiguration von Bild 2.2 nicht dieselben Probleme auf. Im Eingangszustand kann der E/A-Anschluß nur einige Mikroampere liefern.

Wenn man jedoch einen Treiber für die Arbeitsspule wie in Abb. 2.4 benutzt, erkennt der TTL-Eingang der Treiber den hochohmigen Zustand als logische 1 und betätigt die Arbeitsspule. In diesem Fall sind beide Lösungen in Bild 2.3 ausreichend. Die Transistoren sind jedoch mit dem TTL-Puffer verbunden. Außerdem kann der weitere Ausgang, der in Bild 2.3b gezeigt wird, benutzt werden umd die Treiber einzuschalten, wenn ein Einschalt-Eingang zur Verfügung steht. Das wird in Bild 2.5 gezeigt.

2.2.5 Handshaking

Der MCS 6520 bietet die Möglichkeit zur Steuerung von Interrupt und Daten-Übertragung. Das Verfahren zur Steuerung der Übertragung von Daten zwischen Prozessor und einem Peripherie-Baustein wird als Handshaking bezeichnet. Bei diesem Vorgang kann jeder Baustein (Prozessor oder Peripherie) dem anderen anzeigen, daß sein Arbeitsgang zu Ende ist. Die Abfolge ist für Übertragungen in und aus dem Prozessor etwas verschieden. Sie werden daher unten getrennt besprochen.

2.2.5.1 Handshaking für Datenübertragungen vom Prozessor

Aus dem Prozessor werden die Daten durch Einschreiben in das Daten-Register im Peripherie-Baustein MCS 6520 ausgeführt. Diese Daten erscheinen dann auf den Peripherie-Ausgangs-Leitungen und können dort durch das Peripherie-Gerät zur Speicherung, Anzeige usw. gelesen werden. Zur Steuerung dieser Datenübertragung durch Handshaking muß der Prozessor zuerst im Peripherie-Baustein anzeigen, daß am E/A-Kanal Daten verfügbar sind. Das Peripherie-Gerät liest dann diese Daten und zeigt dem Prozessor an, daß die Daten übernommen worden sind und neue Daten verfügbar gemacht werden können. Der Prozessor macht neue Daten verfügbar und der Ablauf wiederholt sich.

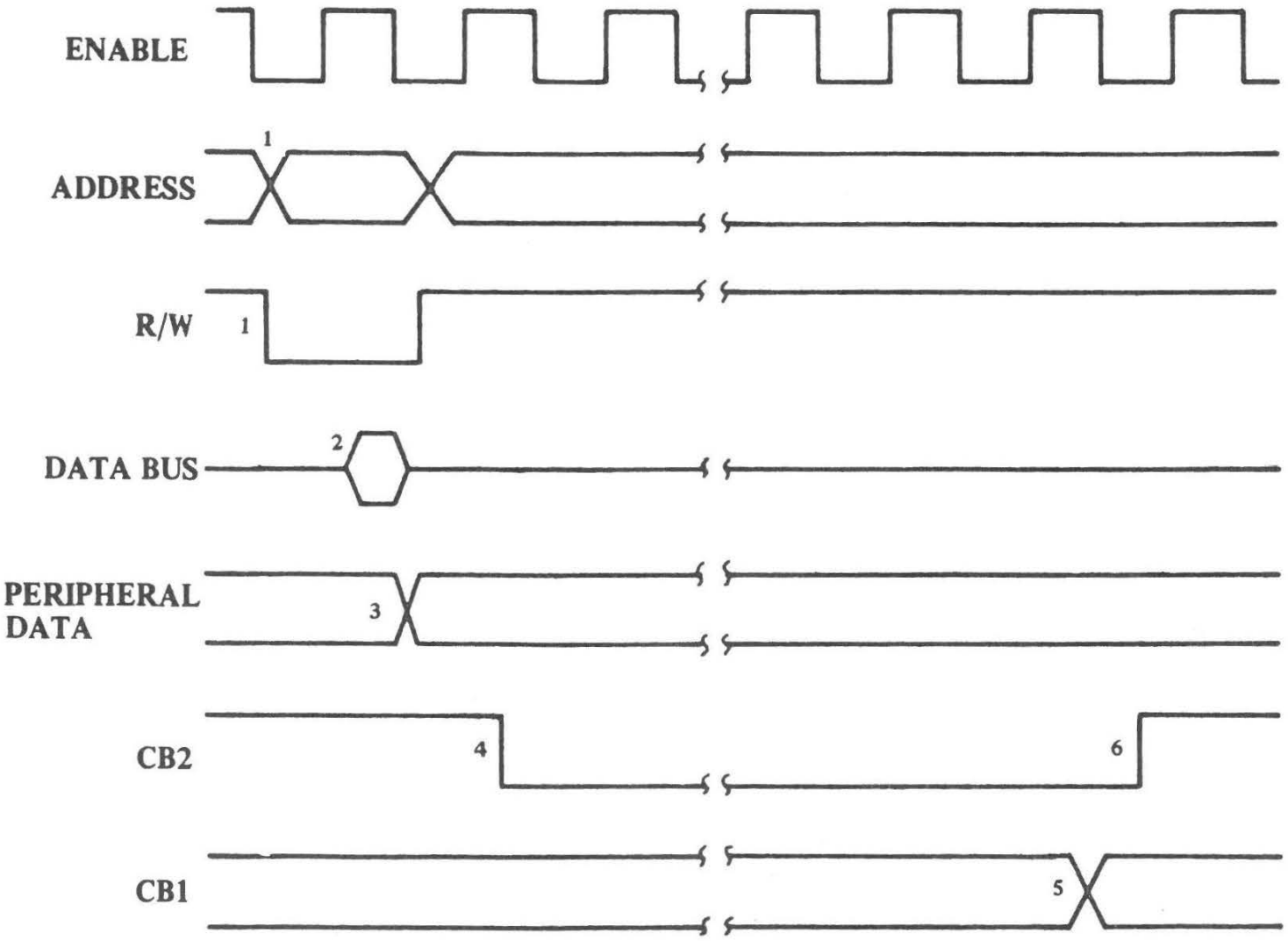


ABBILDUNG 2.6: HANDSHAKE-SEQUENZ FÜR SCHREIBEN

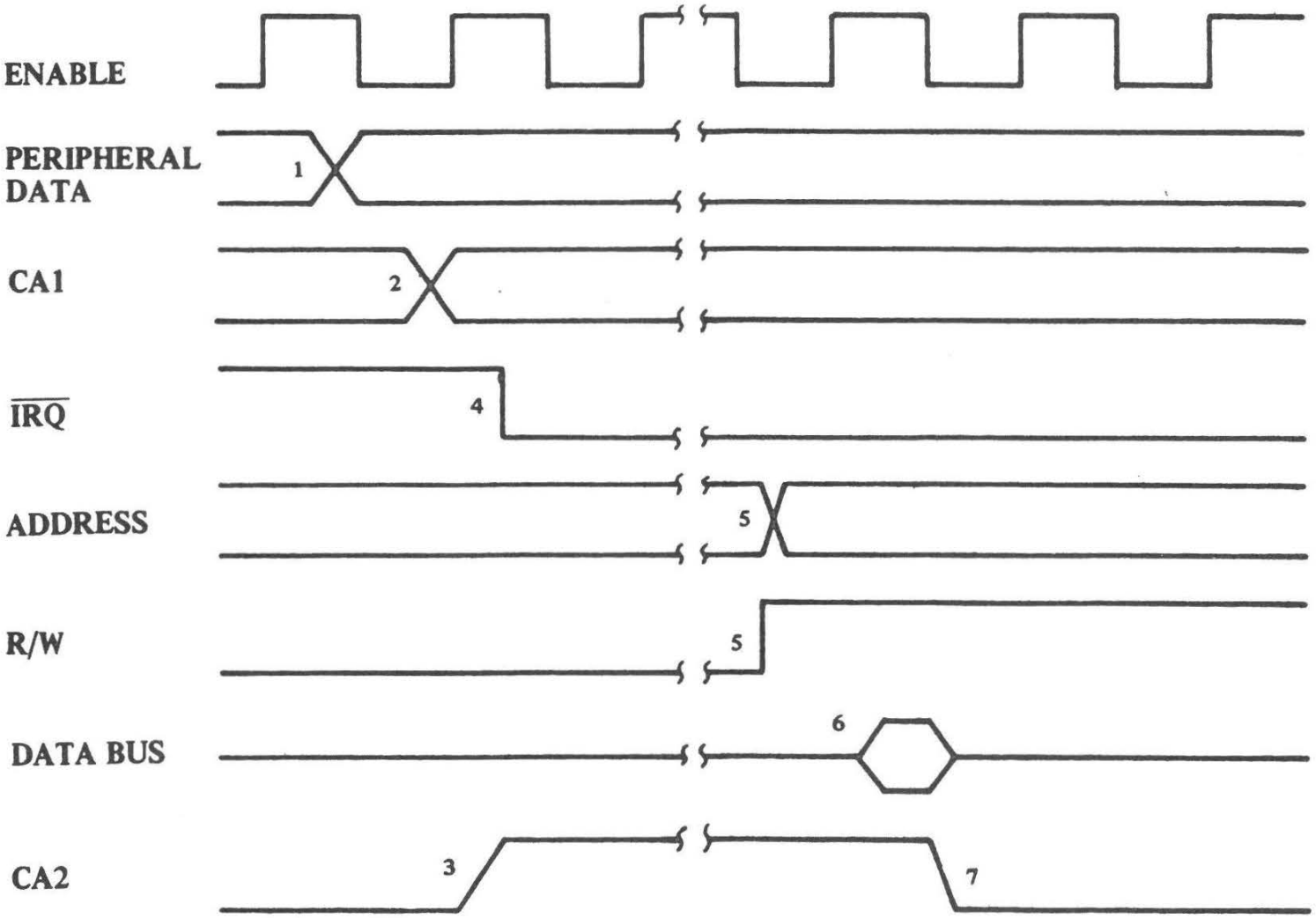


ABBILDUNG 2.7: HANDSHAKE-SEQUENZ FÜR LESEN

Wie in Kapitel 1 beschrieben, kann der Peripherie-Kanal B des MCS 6520 bei Schreib-Vorgängen Handshaking ausführen. Die Peripherie-Steuerleitung kann als Ausgang programmiert werden. Sie geht auf L, wenn der Prozessor Daten in den Peripherie-Kanal B schreibt. Dies Signal zeigt dem Peripherie-Baustein die Verfügbarkeit von Daten auf den Ausgangsleitungen an.

Die Ausgangsleitung CB 2 bleibt auf L, bis der Peripherie-Baustein dem Prozessor meldet, daß die Daten übernommen worden sind. Dazu bekommt der Prozessor einen Interrupt durch den Interrupt-Eingang CB 1. Bild 2.6 zeigt die Abfolge während des Handshaking beim Schreiben.

2.2.5.2 Handshaking bei Datenübertragungen in den Prozessor

Der Peripherie-Kanal A des MCS 6520 ist zum Handshaking bei Datenübertragungen vom Peripherie-Baustein in den Prozessor bestimmt. Bei dieser Reihenfolge muß der Schnittstellen-Baustein dem Prozessor anzeigen, daß Daten verfügbar sind und der Prozessor muß zurückmelden, daß die Daten übernommen worden sind. Das ist im Grunde dieselbe Reihenfolge wie im vorhergehenden Abschnitt. Der Interrupt-Eingang CA 1 zeigt dem Prozessor an, daß auf dem Peripherie-Kanal A Daten verfügbar sind. Der Peripherie-Baustein muß die Daten dann dort halten, bis der Prozessor sie in seine internen Register lädt. Wenn der Prozessor den Peripherie-Kanal A ausliest, geht die Peripherie-Steuerungsleitung CA 2 auf L, um dem Schnittstellen-Baustein zu melden, daß die Daten übernommen worden sind und neue Daten verfügbar gemacht werden können. Der gesamte Ablauf wird in Bild 2.7 gezeigt.

Zur Verbindung mit Bausteinen zur Datenspeicherung oder im allgemeinen jedem Baustein, der Datenblöcke übertragen muß und eine variable Reaktionszeit hat, ist das Handshaking ein sehr wirkungsvolles Verfahren. Wenn der Prozessor die Geschwindigkeit, mit der die Peripherie Daten übernimmt, nicht voraussagen kann, muß ihm die Peripherie zurückmelden, daß sie übernommen hat.

Die Einleitung des Ablaufs zur Datenübertragung wird normalerweise durch einen Satz von E/A-Leitungen erreicht, die vom übertragenden Kanal getrennt sind. Wenn jedoch einmal der Ablauf eingeleitet ist, muß sich der Prozessor mit dem Peripherie-Baustein nur dann beschäftigen, wenn ein Interrupt vorliegt. Dadurch kann der Prozessor das Haupt-Systemprogramm ausführen während er diese Peripherie-Bausteine noch bedient.

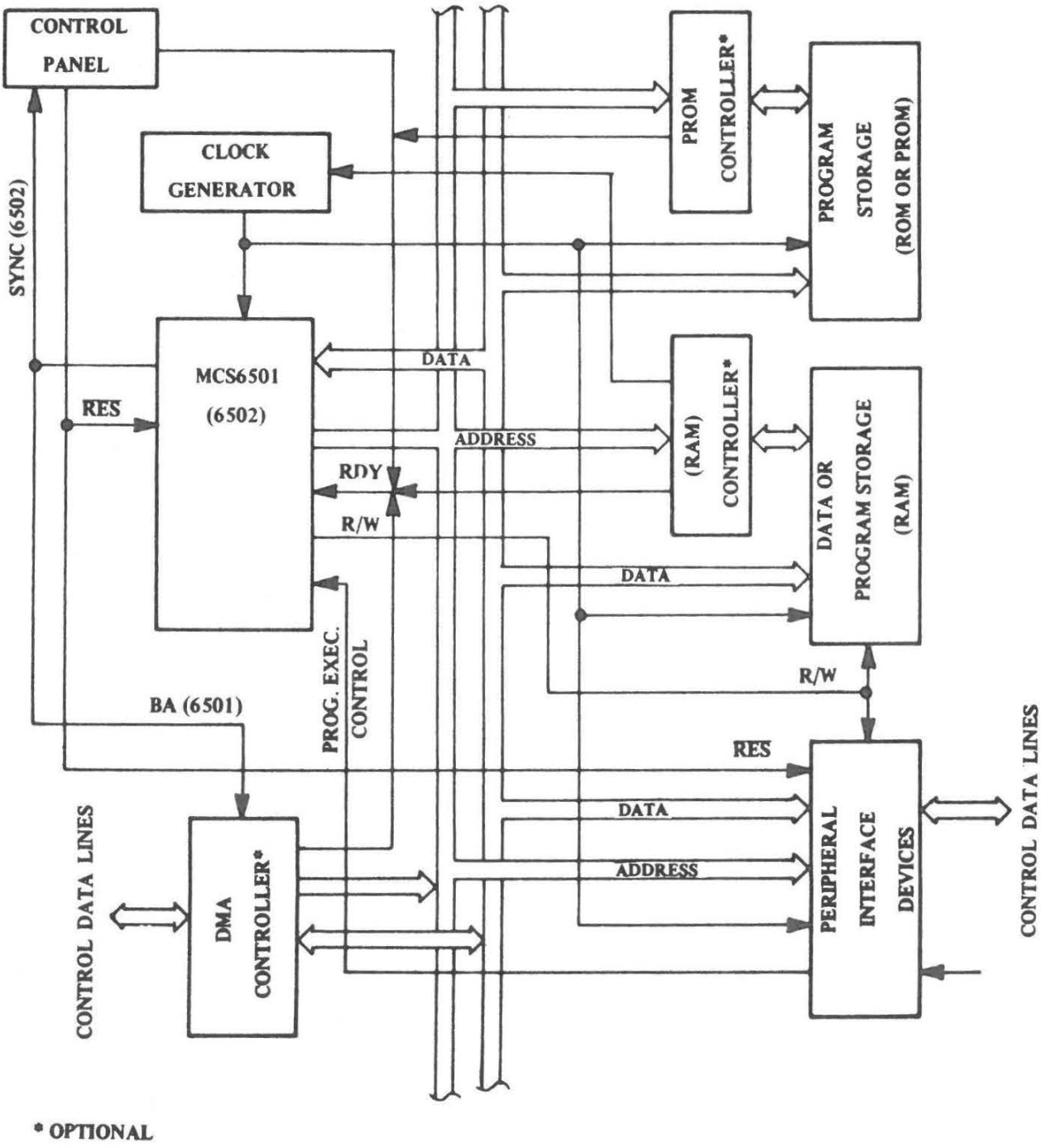
2.3 KONFIGURATION DER SCHNITTSTELLE ZWISCHEN MIKROPROZESSOR UND UNTERSTÜTZUNGS-BAUSTEINEN.

Das Blockschaltbild des Systems (Bild 2.8) zeigt die Datenwege, mit denen das System MCS 6500 arbeitet. Datenbus, Adressbus das Signal R/W usw. sind als einfache Verbindungen zwischen den verschiedenen Bausteinen im System dargestellt. Diese Datenwege bestehen in jedem noch so komplizierten System. Jeder Teil der Mikroprozessor-Schnittstelle muß untersucht werden, ob der Baustein mit Signalen getrieben wird, die alle Daten erfüllen, um sicherzugehen, daß die Zeitverhältnisse zwischen den Bausteinen stimmen und das System im allgemeinen so arbeitet, wie gewünscht.

2.3.1 Zuordnung der Adressen im System MCS 6500

Der Prozessor kann zwischen den verschiedenen RAM, ROM usw. nur durch die Adress-Ausgangsleitungen auswählen. Der Entwickler muß daher diese Leitungen sehr geschickt benutzen, um kleine Systemkosten und eine befriedigende Leistung des Systems zu erreichen.

Bevor nach der günstigsten Konfiguration für die Adressleitungen gesucht wird, um kleinste Systemkosten oder Zeit zur Ausführung des Programms zu erhalten, muß der Entwickler die Beziehung des binären Wertes jeder Adressleitung zum gesamten verfügbaren Adress-Bereich des Mikroprozessors und die Verwendung der UND-Funktion verschiedener Adressleitungen zur Auswahl großer Adress-Blöcke verstehen. Bild 2.9 zeigt den Zustand der drei höchstwertigen Adress-Leitungen für den gesamten verfügbaren Adressbereich des MCS 650X. Die höchstwertige Adress-Leitung liegt für genau die Hälfte des verfügbaren Adress-Bereiches auf logisch 1. Die UND-Funktion der zwei höchstwertigen Adressleitungen liegt für einviertel der verfügbaren Adressen auf logisch 1 usw.



Organization of Microcomputer System

ABBILDUNG 2.8: ORGANISATION DES MIKROCOMPUTER-SYSTEMS

Bild 2.9 zeigt mehrere UND-Funktionen die von den drei höchstwertigen Adressleitungen abgeleitet sind. Jede ist für einen verschiedenen Block der verfügbaren Adressen logisch 1.

Wegen des Baustein-Auswahl-Verfahrens für die Unterstützungs-Bausteine ist die Erzeugung einer UND-Funktion von verschiedenen höherwertigen Adressleitungen sehr wichtig. Der MCS 6520 hat, wie in Kapitel 1, Abschnitt 1.5.2.4 beschrieben, drei Leitungen zur Baustein-Auswahl. Der Baustein wird zum Schreiben oder Lesen von Daten angewählt, wenn CS 1 und CS 2 auf H und CS 3 auf L liegt. Die Auswahl der Adressleitungen, die die verschiedenen Bausteine im System einschalten, ist ein sehr grundlegender, aber wichtiger Teil der System-Konfiguration.

In wenigen Systemen mit Mikroprozessoren ist der Zugriff zu allen 65.536 Worten nötig. Tatsächlich können die meisten Systeme in weniger als 2.000 Worten für Programm und Datenspeicher programmiert werden. Der gesamte Adress-Bereich steht hauptsächlich zur Verfügung, da er die Konfiguration von Systemen mit der kleinsten Anzahl von Bausteinen zur Dekodierung zwischen Prozessor und den Unterstützungs-Bausteinen ermöglicht. Man kann jeder Art von Baustein (RAM, ROM, Peripherie-Schnittstelle usw.) einen Adress-Block im System zuordnen. Diese Zuordnung muß jedoch eindeutig sein. Für jede Adresse im Systemprogramm darf nur ein Unterstützungs-Baustein ausgewählt werden.

2.3.1.1 Zuweisung der Adressen für ROM

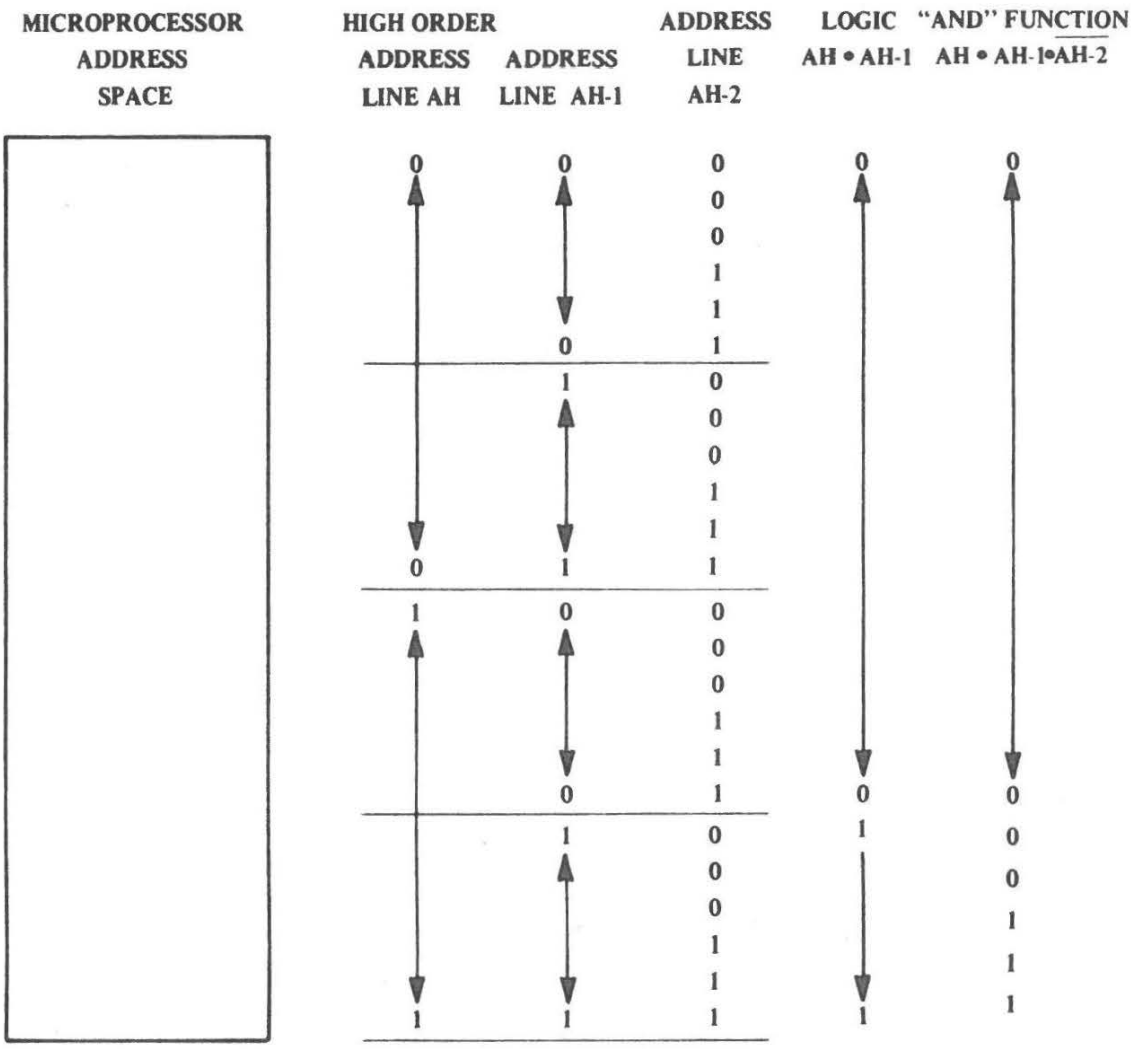
Die Zuordnung der ROM-Adressen wird durch die Tatsache bestimmt, daß die Vektoren für Interrupt und Reset in den sechs höchstwertigen Worten im Speicher liegen müssen. Diese Vektoren liegen fest und müssen dauernd an diesen Stellen gespeichert sein.

Deshalb werden dem Programm-Speicher (normalerweise ROM) die höherwertigen Adressen zugeordnet. Es wird empfohlen, A 15 (A12 für den MCS 6504 und A 11 für MCS 6503 und 6505) zur Auswahl des Programm-ROM zu verwenden.

2.3.1.2 Zuordnung der Adressen für RAM

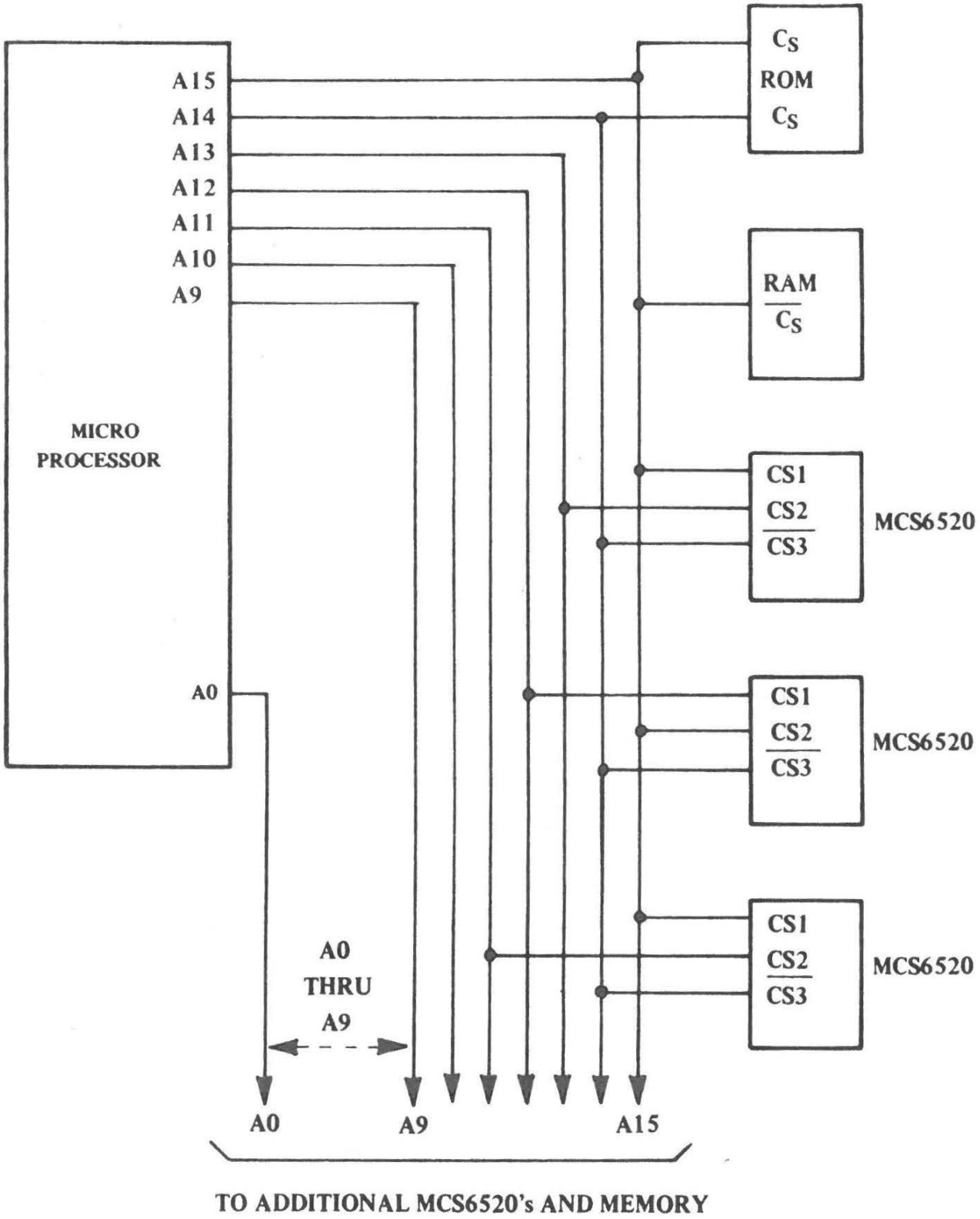
Verschiedene Faktoren bestimmen die Lage des RAM in Systemen mit MCS 650X. Daten, die im Speicher unter Steuerung durch den internen Stapelzeiger des Prozessors gespeichert werden, gehen immer auf Page 1 (ADH = 01). Für alle Adressier-Verfahren der Page 0 muß ein RAM in Page 0 vorhanden sein. Deswegen sollte sich das RAM in einem System mit MCS 650X auf den niederwertigen Adressen im Speicher befinden.

Wenn das RAM im unteren Bereich und das ROM im oberen Bereich des Speichers liegen, müssen die Peripherie-Schnittstellen-Bausteine irgendwo dazwischen liegen. Dies wird in Bild 2.10 dadurch erreicht, daß A 15 · A 14 zur Auswahl von ROM, $\overline{A\ 15}$ zur Auswahl von RAM, und A 15 · $\overline{A\ 14}$ zur Auswahl aller Peripherie-Schnittstellen-Bausteine verwendet wird. So ist eine Unterscheidung zwischen den Arten der Unterstützungs-Bausteine möglich. Die Möglichkeit zur Auswahl jedes Bausteins in den Gruppen vervollständigt die Adress-Struktur. Die Adressen, die die verschiedenen Register, Peripherie-Kanäle usw. in den peripheren Schnittstellen-Bausteinen auswählen, folgen normalerweise nicht aufeinander. Aus diesem Grund wird das in Bild 2.10 dargestellte Verfahren zur Unterscheidung zwischen den Peripherie-Schnittstellen-Bausteinen empfohlen. So können in einem System mit MCS 6502 12 Bausteine ohne Dekodierung, bis zu 9 Bausteinen MCS 6520 in einem System mit MCS 6504 und bis zu 8 Bausteinen in Systemen mit MCS 6503 oder MCS 6505 ausgewählt werden.



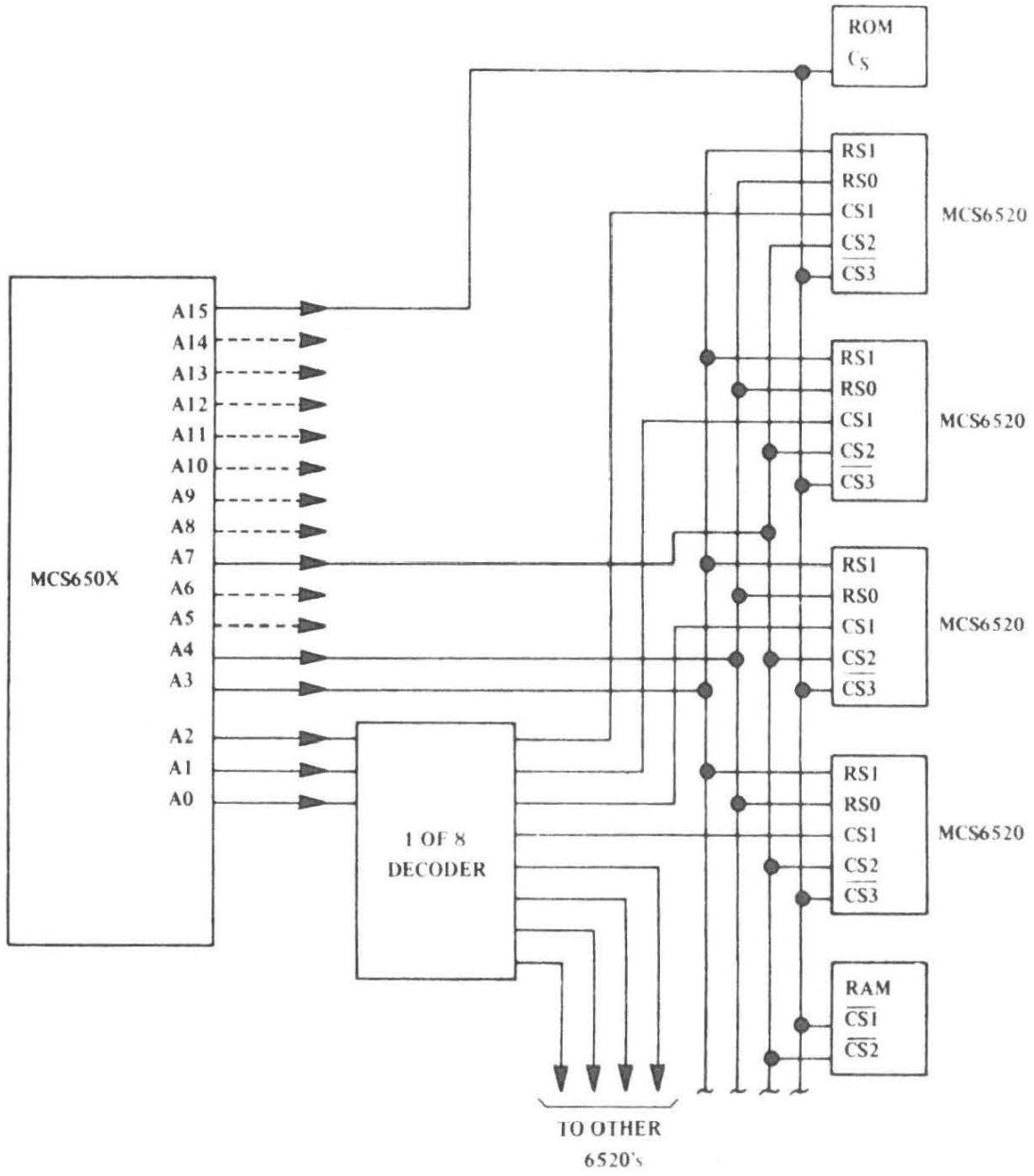
Example of "AND" Function Using High Order Address Lines

ABBILDUNG 2.9: BEISPIEL DER "AND"-FUNKTION UNTER ANWENDUNG DER ADRESSLEITUNGEN



Typical Address Assignments

ABBILDUNG 2.10: TYPISCHE ADRESSZUWEISUNGEN



Page Zero Chip Select Addressing Scheme

ABBILDUNG 2.11: NULL-SEITE, ADRESSIERSCHEMA ZUR BAUSTEIN-AUSWAHL

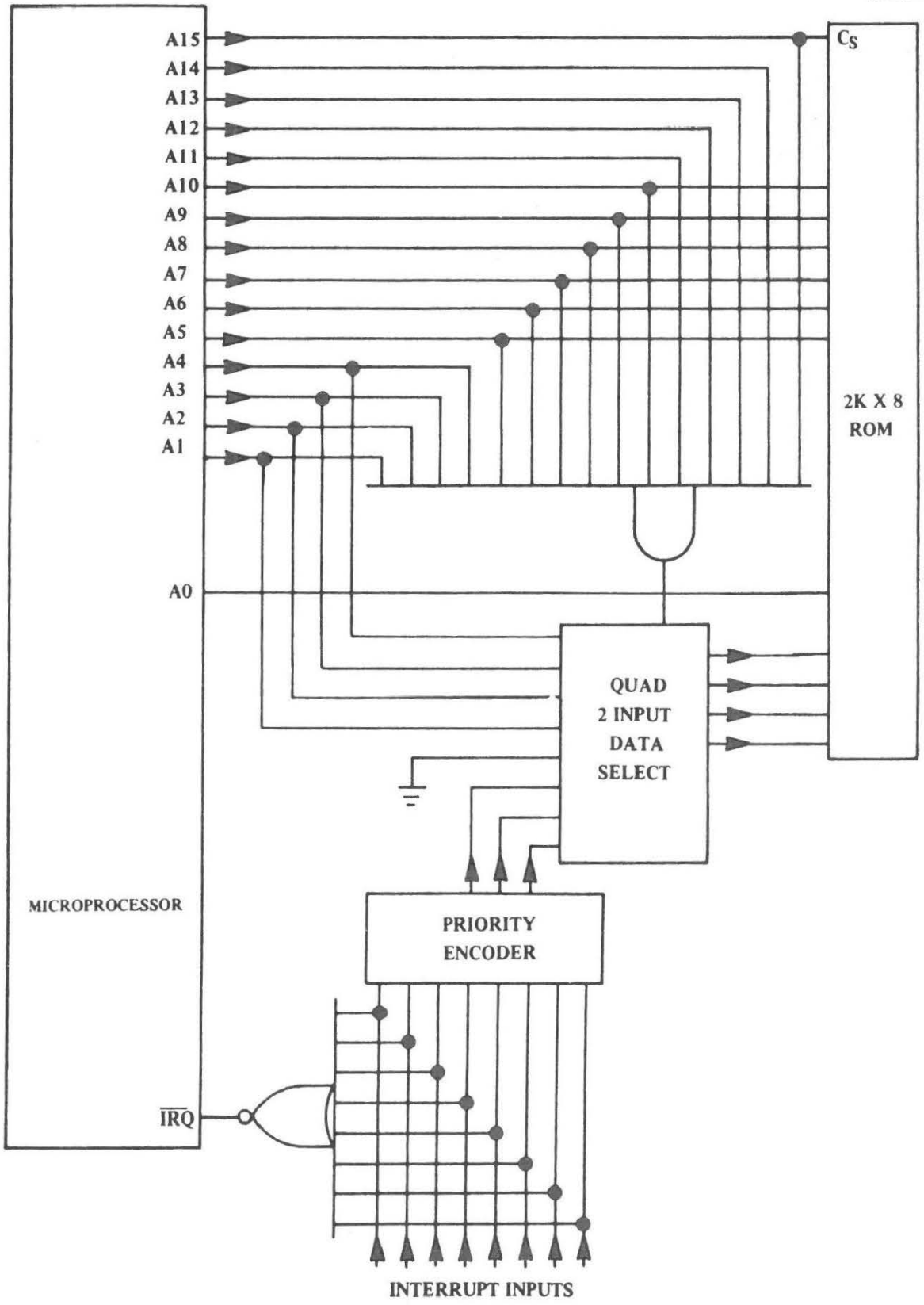
2.3.2 Weitere Verfahren zur Zuordnung von Adressen

In vielen Systemen stellen die oben gezeigten Verfahren nicht die beste Lösung dar. Das gilt besonders, wenn die Geschwindigkeit der Programm-Ausführung an erster Stelle steht.Die Zeit für den Zugriff zu den Peripherie-Bausteinen kann dadurch verkleinert werden, das sie auf Page 0 gelegt werden. Dann kann der gesamte Satz adressierweisen für Page 0 für den Zugriff zu diesen Bausteinen benutzt werden. Die Abfrage der Steuerregister des MCS 6520 während der Bedienung von Interrupts wird sehr erleichtert, wenn die Steuerregister in aufeinander folgenden Adressen liegen. Dann kann zu diesen Registern mit dem indizierten Adress-Verfahren für Page 0 zugegriffen werden, das in der Programmier-Fibel beschrieben ist. Bild 2.11 zeigt die Verdrahtung der Adressen dafür. Es werden externe Adress-Dekoder benötigt, aber die zusätzlichen Kosten werden durch eine höhere Arbeitsgeschwindigkeit gerechtfertigt.

In Mikroprozessor-Systemen ist die Beziehung zwischen Software und Hardware von großer Bedeutung und nur ein volles Verständnis beider erlaubt die Optimierung der Abwägung zwischen Geschwindigkeit und Kosten.

2.3.3 Interrupts

Das Konzept der Interrupts wird in Kapitel 1, Abschnitt 1.3.2 eingeführt. Dort wird jedoch wenig über die Hardware- und Software-Verfahren gesagt, die zur Realisierung des Interrupt-Systems nötig sind. Dieser Abschnitt soll den Entwickler in die Einzelheiten von Interrupt und der Verfahren zur Bedienung von Interrupts einführen.



Selecting the Interrupt Vector

ABBILDUNG 2.12: AUSWAHL DES INTERRUPT-VEKTORS

2.3.3.1 Feststellen der Interrupt-Priorität

Kapitel 1 bezieht sich auf verschiedene Verfahren zur Prioritäts-Festellung von Interrupts durch Hardware zur schnelleren Bedienung der Interrupts. Diese Hardware hat das Ziel, daß der Prozessor direkt in das Programm zur Bediendung des aktiven Interrupts mit der höchsten Priorität geht, ohne Zeit zur Abfrage jedes Bausteins zu brauchen, der einen Interrupt liefern kann.

Alle Hardware-Verfahren zur Feststellung der Priorität beruhen auf dem „ Prioritäts-Encoder " in Bild 2.12. Dieser Baustein hat 8 Eingänge, die Prioritäts-Ebenen von 1 bis 8 zugeordnet sind und erzeugt ein 3 Bit Wort, das das aktive Eingangssignal mit der höchsten Priorität angibt.

Die Erzeugung dieses 3 Bit Wortes ist eine einfache Aufgabe. Seine Verwendung zur Adressierung der Routine zur Bedienung des entsprechenden Interrupt ist erheblich schwieriger und stellt eine Gelegenheit zur Anwendung von Kreativität dar. Hier werden verschiedene Lösungen vorgeführt, um zu zeigen was möglich ist. Es sind sicher nicht die einzigen Lösungen. Jedes System muß getrennt betrachtet werden, um die Optimierung sicherzustellen.

2.3.3.2 Beispiel 1: Auswahl des Interrupt-Vektors

Der letzte Schritt der Interrupt-Antwort im Prozessor ist das Holen eines Interrupt-Vektors von zwei festgelegten Adressen im Speicher. Dieser Interrupt-Vektor zeigt die Adresse der Software an, die der Prozessor zur Abfrage der Bausteine ausführt, die einen Interrupt veranlassen können. Anstatt auf die Routine zur Abfrage zu zeigen, wäre es viel schneller, direkt zu der Software zu gehen, die den Interrupt tatsächlich bedient. Dazu ist ein besonderer Vektor für jeden Interrupt nötig.

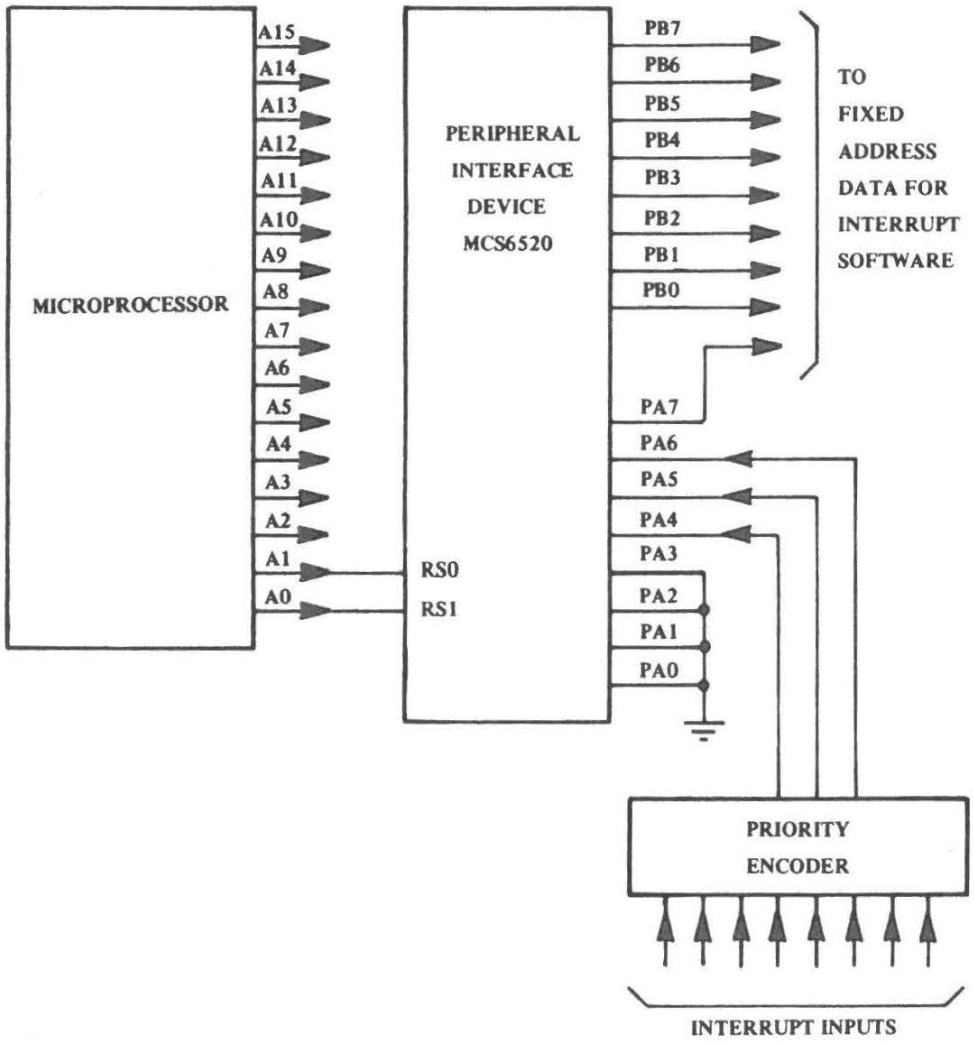
Das in Bild 2.12 gezeigte Verfahren geht davon aus, daß die Interrupt-Vektoren im ROM auf Adressen liegen, die tiefer sind als die normale Adresse des Interrupt-Vektors. Der Decoder erkennt, daß der Prozessor FFFE oder FFFF liest. Zu dieser Zeit werden die Adress-Eingänge AD 1, AD 2 und AD 3 des ROM vom Prioritäts-Encoder getrieben. Statt auf FFFE oder FFFF zuzugreifen, kommt der Interrupt-Vektor von den drei Adressen, die der Prioritäts-Encoder auswählt. Die verwendete Hardware ist recht einfach und die Reaktionszeit für den Interrupt ist so klein wie möglich.

2.3.3.3 Beispiel 2: Benutzung der Software-Möglichkeiten des Prozessors

Die folgenden Lösungen nutzen bestimmt Befehle, die der Prozessor ausführen kann. Die Erste benutzt den Befehl indirekter Sprung (Jump Indirect). Bei diesem Befehl fängt der Prozessor die Ausführung des Programms an, das in den zwei aufeinanderfolgenden Speicherstellen enthalten ist.

Wie in Beispiel 1 wird das 3 Bit Wort vom Prioritäts-Encoder Teil der Adresse der Interrupt-Software. Wenn der Ausgang des Prioritäts-Encoders mit den Eingängen eines Peripherie-Schnittstellen-Bausteins verbunden ist, kann der Prozessor einen indirekten Sprung ausführen, indem er die Adresse an den beiden E/A-Peripherie-Kanälen benutzt. Das wird in Bild 2.13. gezeigt.

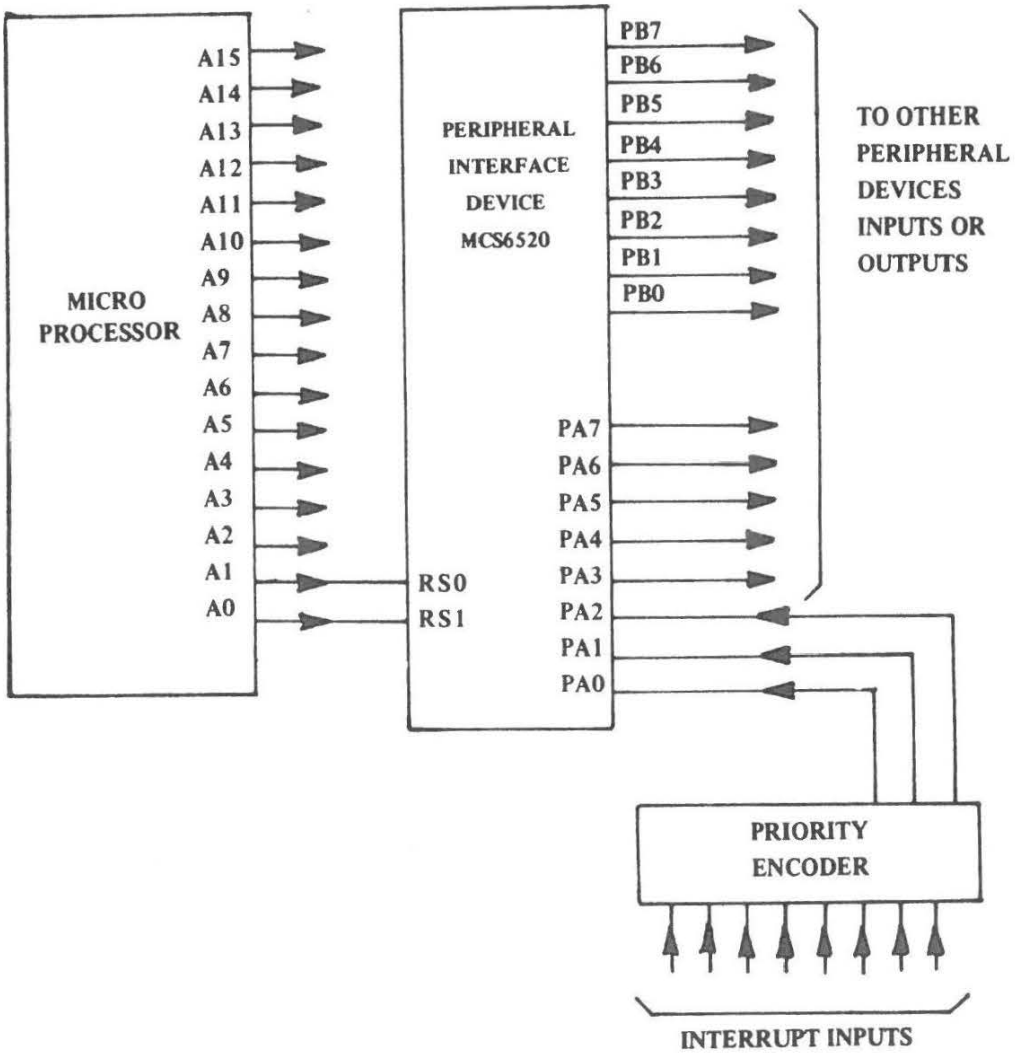
Bild 2.14 zeigt eine andere Lösung, die die Software nutzt. Auch hier ist der Ausgang des Prioritäts-Encoders mit den Eingängen eines E/A-Peripherie-Kanals verbunden. Bei dieser Lösung ist jedoch der Prioritäts-Encoder mit den niederwertigen Bits verbunden und die anderen Bits können zur Steuerung oder als Eingangsleitungen für andere Funktionen verwendet werden.



NOTE: CONNECTING THE ADDRESS LINES AS SHOWN PUTS THE TWO MCS6520 I/O PORTS IN SEQUENTIAL ADDRESSES.

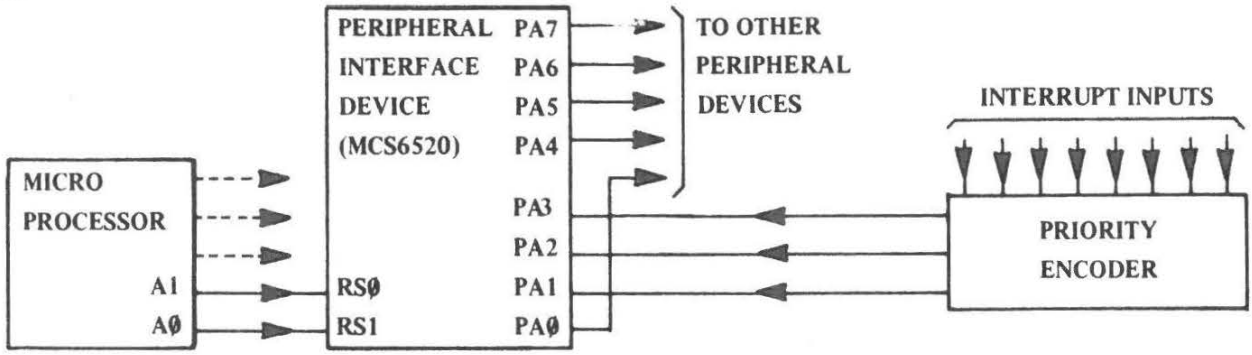
Using MCS6520 for Jump Indirect Interrupt Routines

ABBILDUNG 2.13: ANWENDUNG DES MCS 6520 FÜR INDIREKTE INTERRUPT-ROUTINEN (JUMP)



Priority Encoder Connected to Low Order Bits of MCS6520.

ABBILDUNG 2.14A: PRIORITÄTSENCODER VERBUNDEN MIT DEN UNTEREN BITS DES 6520



Priority Encoder to Peripheral Interface Scheme

ABBILDUNG 2.14B: VERBINDUNG VON PRIORITÄTSENCODER ZUR PERIPHEREN SCHNITTSTELLE

```
INTVEC --->  PHA                    Receive Interrupt Vector
             TXA
             PHA
             LDA    IPA AO          Read PIA Port
             AND    #OE             Clear PIA
             TAX                    Transfer Acc. to X index reg.
             LDA    VEC TAB,X       Load Acc. from Interrupt Vector
                                    Table stored in memory
             STA    JMP1            Set Low Order Address Byte
                                    of Interrupt Vector
             INX                    Increment X Index Register
             LDA    VEC TAB,X       Load Acc. from Interrupt Vector
                                    Table
             STA    JMP1+1          Set high order Address Byte
                                    of Interrupt Vector
             JMP    (JMP1)          Go to Interrupt Service
                                    Software
```

Software Program to Implement Interrupt from above Hardware Configuration

ABBILDUNG 2.15: SOFTWARE-PROGRAMM ZUM IMPLEMENTIEREN DES INTERRUPTS IN OBIGER HARDWARE-KONFIGURATION

Bei dieser Methode werden die 3 Bits vom Prioritäts-Encoder Teil einer im Speicher festgelegten Adresse. Diese Adresse wird dann wie oben im Befehl indirekter Sprung benutzt. Bild 2.15 zeigt Einzelheiten.

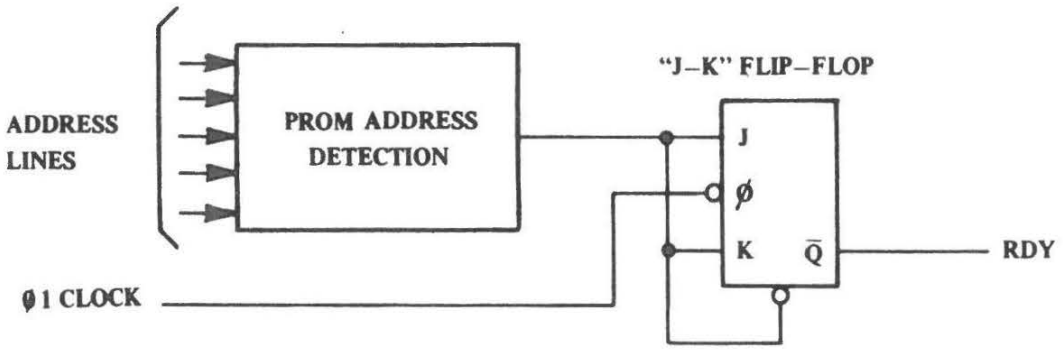
2.3.4 Die Anwendung von RDY zur Steuerung der Speicher-Schnittstelle

Bei der Verwendung von Speicher-Bausteinen, die nicht direkt mit der Familie MCS 650X kompatibel sind, ist die Möglichkeit, den Prozessor anzuhalten sehr wichtig.

Das Signal RDY beim MCS 6502 und 6505 kann benutzt werden, den Prozessor in jedem Zyklus anzuhalten, in dem er nicht versucht Daten in den Speicher einzuschreiben. Der Prozessor kann für eine beliebige Zahl von Takt-Zyklen angehalten werden. Das kann einen Zyklus dauern für die Verbindung mit langsamen Speichern bis zu vielen Zyklen für DMA-Anwendungen und für die Ausführung eines Einzel-Zyklus.

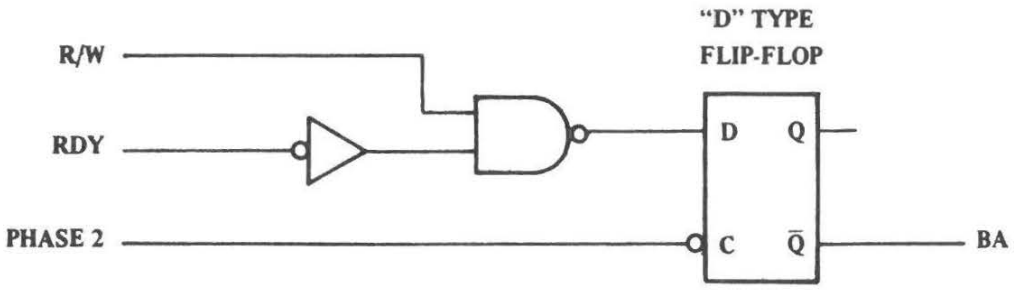
2.3.4.1 Schnittstelle mit langsamen PROM

Ein der Hauptanwendungen von RDY ist die Steuerung von UV-PROM oder EAROM. Diese Bausteine haben im allgemeinen längere Zugriffszeiten, als der Prozessor erfordert, wenn er mit einer Taktfrequenz von 1 MHz arbeitet. Sie können innerhalb 100 ns vom Ende des Taktes 2 keine Daten auf den Datenbus geben. Der Takt 2 wird zum Speichern von Daten oder Befehlen auf dem Datenbus benutzt. Daher muß der Prozessor für einen ganzen Zyklus angehalten werden, wenn die Daten nicht zur richtigen Zeit verfügbar sind. Der Befehl wird dann beim folgenden Takt 2 gespeichert. Die Ausführung des Befehls geht dann beim nächsten Zyklus weiter. In Bild 2.16 wird eine Logik zur Ausführung dieser Funktion vorgeschlagen.



Interfacing Scheme for Slow PROM's

ABBILDUNG 2.16: SCHNITTSTELLE FÜR LANGSAME PROMS



Logic Used to Generate Bus Available Signal for DMA Applications

ABBILDUNG 2.17: LOGIK ZUM ERZEUGEN DES SIGNALS "BUS VERFÜGBAR" FÜR DMA-ANWENDUNGEN

Die Daten die bei der Ausführung des Befehl im folgenden Zyklus verwendet werden, sind diejenigen, die während des Taktes 2 nachdem RDY auf H gegangen ist auf dem Datenbus liegen.

2.3.4.2 Verfahren zum direkten Speicher-Zugriff (DMA, Direct Memory Access

Die Übertragung von Daten aus Peripherie-Speichergeräten in den Speicher des Mikrocomputers kann normalerweise Byte für Byte unter Steuerung des Prozessors vor sich gehen. In großen Datensichtgeräten, Steuerungssystemen usw. kann der Hauptspeicher ein Band-oder Plattenspeicher mit hoher Geschwindigkeit sein. In solchen Systemen muß die Datenübertragung vom Speicher-Gerät in den Speicher mit Geschwindigkeiten erfolgen, die zu groß für den Prozessor sind. Die Steuerung der Übertragung muß außerhalb des Prozessors durch einen getrennten Controller erfolgen und das Peripherie-Gerät muß einen direkten Zugriff zum RAM des Systems haben.

Der direkte Speicher-Zugriff erfordert hauptsächlich, daß der Prozessor nicht zum Speicher zuzugreifen braucht. Das wird im allgemeinen durch Anhalten des Prozessors sichergestellt. Die DMA-Steuerung muß dann die Leitung R/W und die Busse für Adresse und Daten der Speichereinheit übernehmen.

Beim MCS 6502 und 6505 besteht die Möglichkeit zum Anhalten des Prozessors. Dazu wird die Leitung RDY am Prozessor auf L gelegt. Der Prozessor hält im ersten Zyklus an, in dem er nicht schreibt. Der Datenbus ist im hochohmigen Zustand. Nach dem Anhalten des Prozessors muß die DMA-Steuerung die Adresse und Daten für liefern und R/W steuern, wenn Daten in den Speicher übertragen werden. Adressen für die Speicher können entweder von der DMA-Steuerung oder dem Prozessor durch Gatter aufgeschaltet werden. Das kann auf einfache Weise mit einem vierfachen Datenwähler mit zwei Eingänge erreicht werden.

Während des DMA-Betriebes liefert die DMA-Steuerung die Adressen. Wenn der DMA-Betrieb fertig ist, wird das Signal zur Auswahl des Eingangs am Datenwähler umgekehrt und die vom Prozessor erzeugten Adressen bestimmen wieder zu welchem Speicherwort zugegriffen wird. Die Leitung R/W zu den Speichern kann auf dieselbe Weise wie die Adress-Leitungen gesteuert werden.

Der Datenbus muß auf eine etwas andere Weise gesteuert werden. Das wird durch die Tatsache nötig, daß diese Leitungen in zwei Richtungen arbeiten. Die Datenbus-Anschlüsse am Prozessor und den Unterstützungs-Bausteinen dienen sowohl als Ein- als auch als Ausgang. Die Ausgangspuffer in jedem Baustein können in den hochohmigen Zustand gehen, damit ein beliebiger Baustein den Bus während der Übertragung von Daten oder Befehlen treiben kann. Aus diesem Grund ist zur Verbindung der DMA-Steuerung mit dem System-Datenbus ein Bus-Erweiterungs-Baustein für zwei Richtungen und mit drei Ausgangszuständen nötig. Bild 2.17 zeigt die zur vollständigen Steuerung von Adressbus und Datenbus bei DMA-Anwendungen nötige Logik. Sie erzeugt das Signal „Bus verfügbar" (Bus Available) für die DMA-Steuerung.

2.3.4.3 Steuerung von dynamischen RAM im System MCS 6500

Für Systeme mit einem großem Schreib/Lese-Speicher (RAM) kann das dynamische RAM mit 4096 Bit den erforderlichen Speicherplatz mit der kleinsten Zahl an Bausteinen liefern. Derzeit lieferbare dynamische RAM können vier Mal so viel Daten wie ähnliche statische Bausteine speichern. Diese Bausteine haben jedoch einen großen Nachteil - sie müssen regelmäßig aufgefrischt werden (Refresh). Die Refresh-Periode ist für die lieferbaren Bausteine etwa bei 2 ms für den ganzen Baustein. Der Refresh des ganzen Bausteins erfordert 32 Lesevorgänge, die alle zusammen alle zwei ms durchgeführt werden können oder je einer alle 64 Mikrosekunden.

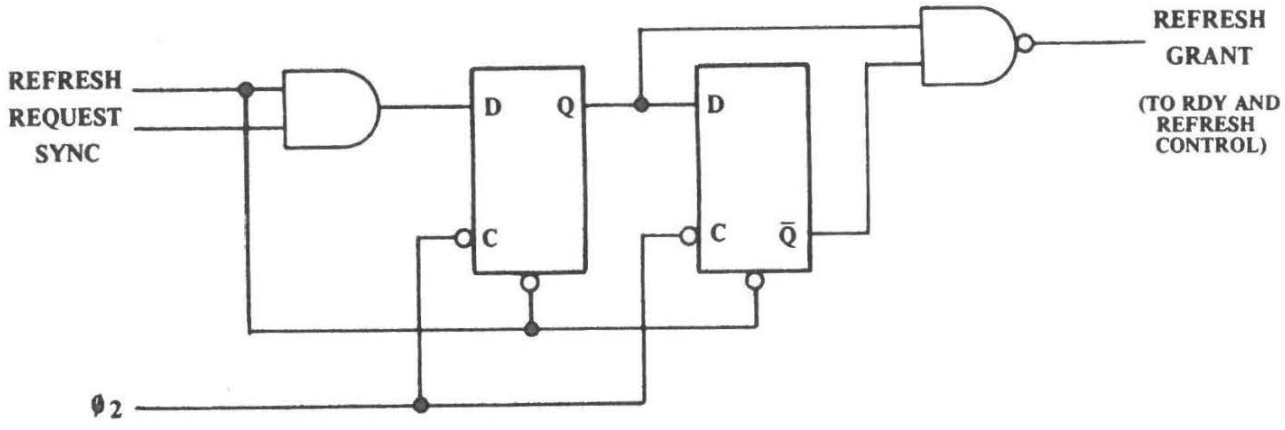
Die Verwendung dynamischer Speicher kann für die Leistung des Systems sehr nachteilig sein, wenn keine getrennte Steuerung zur Ausführung des Refresh-Vorgangs verwendet wird.

Wie bei jedem direkten Speicherzugriff muß der Prozessor angehalten werden, damit nicht Prozessor und DMA-Steuerung zur gleichen Zeit den Zugriff zum Speicher versuchen. Der Eingang RDY bietet diese Möglichkeit. Ein direkt vom Systemtakt betriebener Zähler stellt ein sehr passendes Refresh-Signal dar. Jedes Mal, wenn der Zähler durch den Zählerstand 63 geht gibt es einen Impuls zur Anforderung eines Refresh. Der tatsächliche Refresh-Vorgang für den Speicher muß während eines Lese-Vorgangs stattfinden, wenn der Prozessor für einen Zyklus angehalten ist. Es wird auf gleiche Weise wie bei DMA- Vorgängen festgestellt, wann der Prozessor angehalten ist. Die Steuerung muß die Leitung RDY auf L ziehen und dann die Leitung R/W untersuchen, um festzustellen wann der Prozessor in einem Lese-Zyklus ist.

Der genaue Vorgang während eines Refresh-Zyklus hängt von den verwendeten Bauteilen ab. Die Zeit für den Refresh des Speichers ist „ N - 1/2 " Zyklen, wobei N die Zahl der Zyklen bedeutet, in denen der Prozessor angehalten ist. Der erste halbe Zyklus geht verloren, weil der Zustand der Leitung R/W nicht vor Takt 2 als gültig angesehen werden kann. Die Steuerung der Speicher-Adressleitungen muß zum Beginn von Takt 1 an den Prozessor zurückgegeben worden sein, wenn die Speicher einen vollen Zyklus haben sollen, um gültige Daten auf den Datenbus zu geben.Das läßt zur Ausführung des Refresh-Vorgangs einen halben Zyklus übrig, wenn der Prozessor für einen Zyklus anhält. Durch Anhalten des Prozessors für zwei Zyklen können 1,5 Zyklen zur Verfügung stehen. Diese Möglichkeit ist mit den meisten zur Zeit lieferbaren RAM kompatibel.

Das Hauptproblem bei der Realisierung von Systemen mit dynamischen RAM ist es, zu wissen, wann der Prozessor angehalten ist. In den Realisierungen, die oben beschrieben sind, wird dazu ein halber Zyklus benötigt. Der MCS 6502 liefert jedoch ein Signal, mit dem vorhergesagt werden kann, das der Prozessor genau im nächsten Zyklus anhält. Das ist das Signal SYNC. Einem Zyklus, in dem ein Befehl geholt wird, kann auf keinen Fall sofort ein Schreibvorgang folgen. Damit kann die Steuerung zum Speicher-Refresh die Steuerung der Adressleitungen zum Anfang des Zyklus übernehmen statt nach der abfallenden Flanke des Taktes 1.

Die Leitung RDY wird bei Takt 1 auf L gezogen und der Prozessor hält sicher an. Beim nächsten Takt 1 wird dem Prozessor die Steuerung der Adressleitungen zurückgegeben und RDY wird zur gleichen Zeit auf H gesetzt. Als Ergebnis verliert der Prozessor nur einen Zyklus, während die Refresh-Logik einen ganzen Zyklus zum Refresh der Speicher zur Verfügung hat. Ein Vorschlag für diese Steuer-Logik wird in Bild 2.18 gezeigt.



Control Logic for Refresh Signal for Dynamic RAMS

ABBILDUNG 2.18: STEUER-LOGIK FÜR REFRESH-SIGNAL FÜR

2.4 WEITERE SYSTEM-BETRACHTUNGEN

Vor der endgültigen Festlegung des Entwurfs und nachdem die grundlegende System-Konfiguration vollständig ist, ist normalerweise ein Probeaufbau und seine Prüfung nötig. Dem sollte aber eine genaue Bewertung der Kosten und Leistung des vorgeschalgenen Entwurfs vorausgehen, um sicherzugehen, daß die verschiedenen Ziele des Projektes erreicht werden.

Der erste Schritt in der Bewertung des Entwurfs ist die Abschätzung des nötigen ROM und RAM und der Zahl und Art der Schnittstellen-Bausteine zur Steuerung der Peripherie.

2.4.1 Peripherie-Schnittstellen-Bausteine

Zahl und Art der Peripherie-Bausteine können im allgemeinen sehr genau abgeschätzt werden.Nach genauer Untersuchung der Leistung des Systems sollten diese Schätzungen einer erneuten Oberprüfung unterzogen werden.Es kann zum Beispiel nötig sein einen speziellen Schnittstellen-Baustein zu verwenden oder die E/A-Struktur neu zu entwerfen, wenn sich zeigt, daß das System nicht mit der erforderlichen Geschwindigkeit arbeiten kann. Die Anwendung von speziellen Peripherie-Schnittstellen-Bausteinen verkleinert die Zahl der Aufgaben, die der Prozessor ausführen muß. So kann im allgemeinen für zusätzliche Bauteilkosten die Geschwindigkeit des Gesamtsystems erhöht werden.

Ebenso kann der ausschließliche Gebrauch von Interrupt-Vektoren zu verbesserter Leistung bei größeren Kosten führen. Das Ziel jedes Entwicklungsprogramms ist die Erfüllung der nötigen Systemleistung zu kleinsten Kosten.

Durch Untersuchung der verschiedenen Peripherie-Geräte im System wird bestimmt wieviele Ein- und Ausgänge gebraucht werden. Die Summe kann durch 16 geteilt werden,um die Zahl der Peripherie-Bausteine festzustellen. Das ist eine gute erste Näherung, die bei weiterem Fortschritt der Entwicklung des Systems erneut überprüft wird.

2.4.2 RAM

Die Abschätzung der erforderlichen Menge von RAM im System ist etwas schwieriger als bei den Peripherie-Bausteinen. Das liegt hauptsächlich daran, daß ein großer Teil dieses RAM von der System-Software als Arbeitsspeicher benutzt wird, z.B. zur Speicherung der Ergebnisse in arithmetischen Operationen. Da das Systemprogramm wahrscheinlich bei dieser ersten Abschätzung noch nicht geschrieben ist, ist die Wahrscheinlichkeit eines Irrtums in diesem Abschnitt der Schätzung recht hoch.

Zusätzlich zum Arbeitsspeicher muß das RAM Speicher stellen für:

1. Den Stapel, genauer in der Programmier-Fibel beschrieben.
2. Speicherung der Peripherie-Eingangs-Daten.
3. Speicherung der Peripherie-Ausgangs-Daten.

Punkt 2 und 3 können recht genau abgeschätzt werden, da die Peripherie-Geräte normalerweise schon genau untersucht worden sind, wenn diese Schätzung zuerst versucht wird. Im allgemeinen muß für jeden Peripherie-Baustein ein Block des RAM zur Verfügung stehen. Die Menge des nötigen RAM hängt von der Art des Peripherie-Bausteins ab und wie der Baustein gesteuert wird.

Die Menge des für den Stapel nötigen RAM hängt sowohl von der Interrupt-Struktur als auch der System-Software ab. Die Schätzung der Erfordernisse durch den System-Programmierer gibt die Grundlage dafür.Zusammen mit einer Schätzung des nötigen Arbeitsspeichers und der Peripherie-Daten-Speicherung erhält man eine Schätzung für das RAM im Gesamtsystem.

2.4.3. ROM

Die nötige Menge ROM im System kann erst nach Vollendung des System-Programms genau festgestellt werden. Durch Teilung des System-Programms in definierbare Abschnitte kann für jede Aufgabe eine Abschätzung getroffen werden und das erforderliche ROM kann damit bestimmt werden.

Die meisten Programme bestehen aus leicht zu definierenden Abschnitten wie der Software für jedes Peripherie-Gerät, arithmetischen Routinen usw. Diese Teile sollten getrennt untersucht werden, um das von ihnen erforderte ROM abzuschätzen.

2.5 UNTERSUCHUNG DER LEISTUNG DES SYSTEMS

Die Struktur der peripheren Schnittstelle in einem System ist recht einfach zusammenzustellen, wenn man von der Verwendung von MCS 6520 ausgeht. Die Leistung des gesamten Systems muß untersucht werden, bevor man mit dem Bau der Hardware zu weit fortgeschritten ist, um zu verhindern, daß während der späteren Stufen des Entwurfs größere Probleme auftreten.

Zunächst wird bestimmt, ob der Prozessor alle Interrupts mit der geforderten Geschwindigkeit verarbeiten kann und dann ob der Prozessor genügend Zeit zur Ausführung von anderen Arbeitsgängen hat. Die Interrupt-Struktur geht davon aus, daß zeitweise mehr als ein Interrupt vorliegt und es Verzögerung in der Bedienung einiger Interrupts durch das Vorhandensein anderer Interrupts gibt. Diese Struktur arbeitet zufriedenstellend, wenn diese Verzögerungen nicht zu groß sind.

Die Zeit zur Verarbeitung der Interrupts sollte beginnend mit dem Interrupt höchster Priorität, dann mit der nächst niedrigeren Priorität usw. untersucht werden. Dabei ist auf die gesamte Zeit zu achten, die wegen gleichzeitiger Interrupt höherer Priorität verloren werden kann. Bei jeder Untersuchung eines Interrupts muß die längste Reaktionszeit des Prozessors abgeschätzt werden. Wenn diese Zeit für die durch den Interrupt ausgeführte Funktion ausreicht, kann man erwarten, daß dieser Teil der Arbeitsweise des Systems eine ausreichende Leistung bringt.

Die Fähigkeit der Mikroprozessoren MCS 650X, Interrupts schnelle und einfach zu verarbeiten, stellt eine der wirklichen Stärken dieser Mikroprozessor-Familie dar.

Es ist jedoch in jedem zu entwickelnden System wichtig, daß der Teil der Zeit, den der Prozessor zur Bedienung von Interrupts benötigt, nicht so lang ist, daß die interne Handhabung von Daten, arithmetischen Opertationen usw. nicht mehr richtig ausgeführt werden können.

Da die Interrupts normalerweise asynchron sind und keine direkte Beziehung zum Hauptprogramm haben, kann man die für die Interrupts verwendete Zeit als durchschnittlichen Prozentsatz der Gesamtzeit betrachten. Die Geschwindigkeit, mit der das Hauptprogramm ausgeführt wird, wird durch diesen Prozentsatz verkleinert.

KAPITEL 3

INBETRIEBNAHME DES MICROCOMPUTERSYSTEMS MCS 6500

3.0 EINFÜHRUNG IN DIE PRÜFUNG VON MICROCOMPUTERN

Nach vielen Stunden der Planung, des Aufbaus und der Programmierung stellt sich dem Microcomputer-Systementwickler seine vielleicht schwierigste Aufgabe: die Inbetriebnahme seines Systems. Der moderne Microcomputer mit einen wenigen Bausteinen und seiner geringen Zahl von Steuer- und Datenleitungen stellt einen großen Fortschritt im Systementwurf dar, wenn alles einwandfrei arbeitet. Für den Entwickler, der die Fehlersuche in Hard- und Software, aus denen ein System besteht, versucht, kann die Prüfung sich zu einem Alptraum entwickeln.

Einem Microcomputer fehlen viele Dinge, die die Prüfung herkömmlicher Logik verhältnismäßig einfach machen. Einmal kann man die Steuersignale, Datenübertragung usw., mit denen das System arbeitet einfach nicht sehen. Außerdem ist es unmöglich den Inhalt der Register, die die Daten im Prozessor speichern, direkt zu untersuchen. Die Untersuchung dieser Daten ist nur indirekt möglich durch Betrachtung der Signale an Ein- und Ausgängen zur rechten Zeit.

Kompliziert wird dieses Problem dadurch, daß viele Programme „dynamisch" geprüft werden müssen. D.h. das System muß mit seiner vollen Arbeitsgeschwindigkeit laufen, unter Umständen ohne wiederkehrende Ereignisse oder brauchbare Triggersignale für ein Oszilloskop.

Es ist aus diesem und vielen anderen Gründen wichtig, daß der Systementwickler wirkungsvolle Prüfungsmöglichkeiten sowohl in Hardware als auch in Software einbaut. Das gilt besonders für Prototypen.

Diese Maßnahmen zusammen mit den Verfahren , die weiter unten besprochen werden, verkürzen Zeit und Aufwand zur Herstellung des ersten arbeitenden Systems. Nachdem Programm und Hardware vollständig fehlerfrei sind, können viele Prüfwerkzeuge, die unten besprochen sind, aus dem Prototypentwurf entfernt werden, ohne die Wirkung des Systems zu beeinträchtigen. So kann der Entwickler, kurze Zeit nachdem die Prototypen zufriedenstellend arbeiten, zu seinem endgültigen Produktionsentwurf gelangen.

3.1 STATISCHE PRÜFUNG

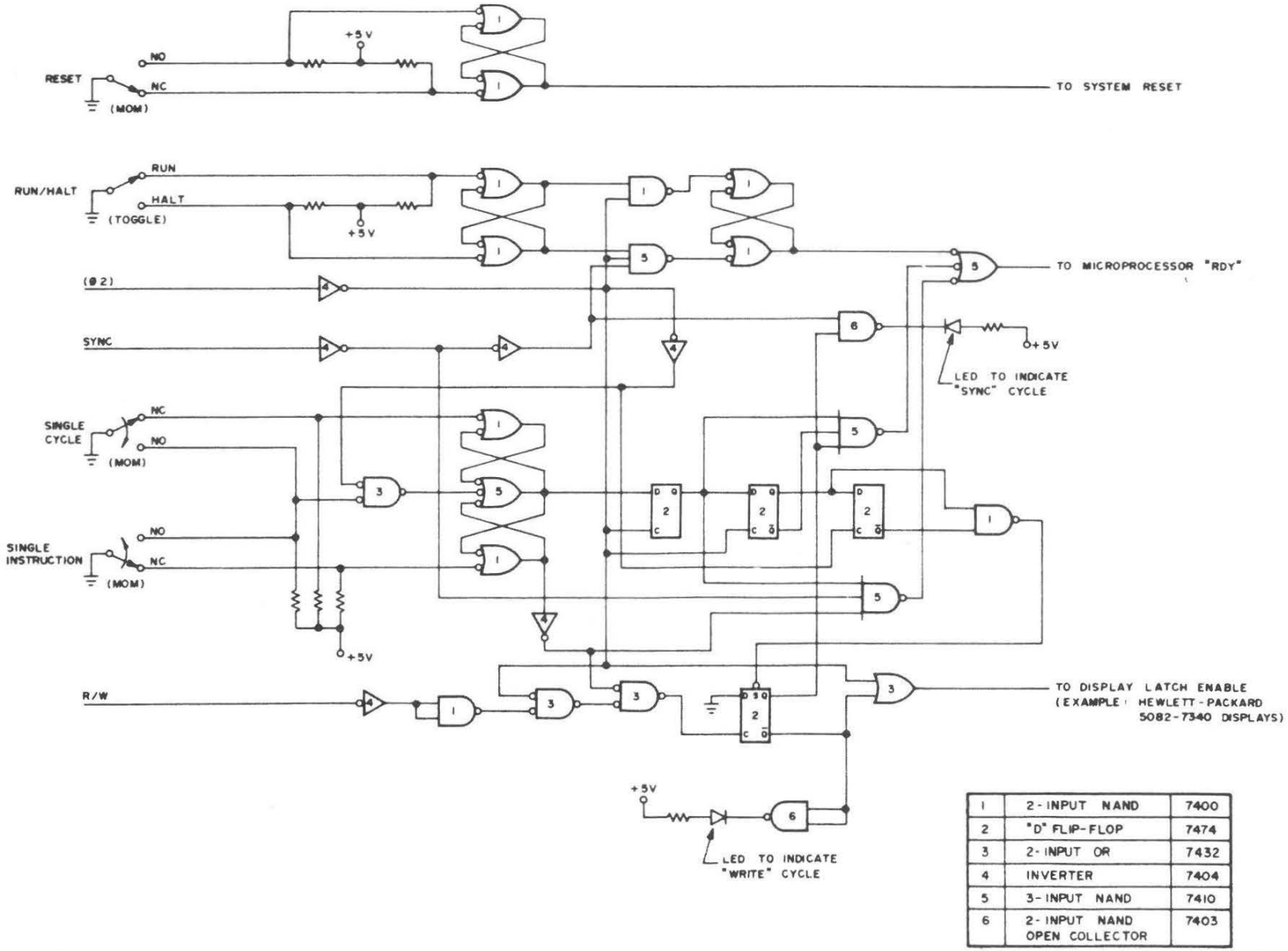
3.1.1 Einführung

Der erste Schritt in der Prüfung jedes Systems ist die statische Prüfung, d.h. Ausführung des Programms mit einem Zyklus oder einem Befehl zur Zeit. Auf diese Weise kann der allgemeine Fluß des Programms untersucht werden und für den größten Teil des Programms die Richtigkeit der Datenübertragung in und aus dem Speicher bestätigt werden. Die Logik zur Steuerung des RDY, Bild 3.1, zur Ausführung eines einzelnen Zyklus und eines einzelnen Befehls ist verhältnismäßig einfach. Diese Schaltung und ihre Verwendung zur Systemprüfung werden unten besprochen.

3.1.2 Ausführung eines Einzelzyklus

Bild 3.2. zeigt das Zeitdiagramm für die Ausführung eines Einzelzyklus. Die Leitung RDY ist L (0 Volt) um den Prozessor anzuhalten. Zur Ausführung eines einzelnen Zyklus geht jedesmal, wenn der Schalter betätigt wird, die Leitung RDY für einen Zyklus auf H (+2,4 V). Die Leitung RDY geht hoch, während der Takt Ø 1 hoch liegt. Der interne Zeitzähler schaltet beim nächsten Taktimpuls Ø1 fort.

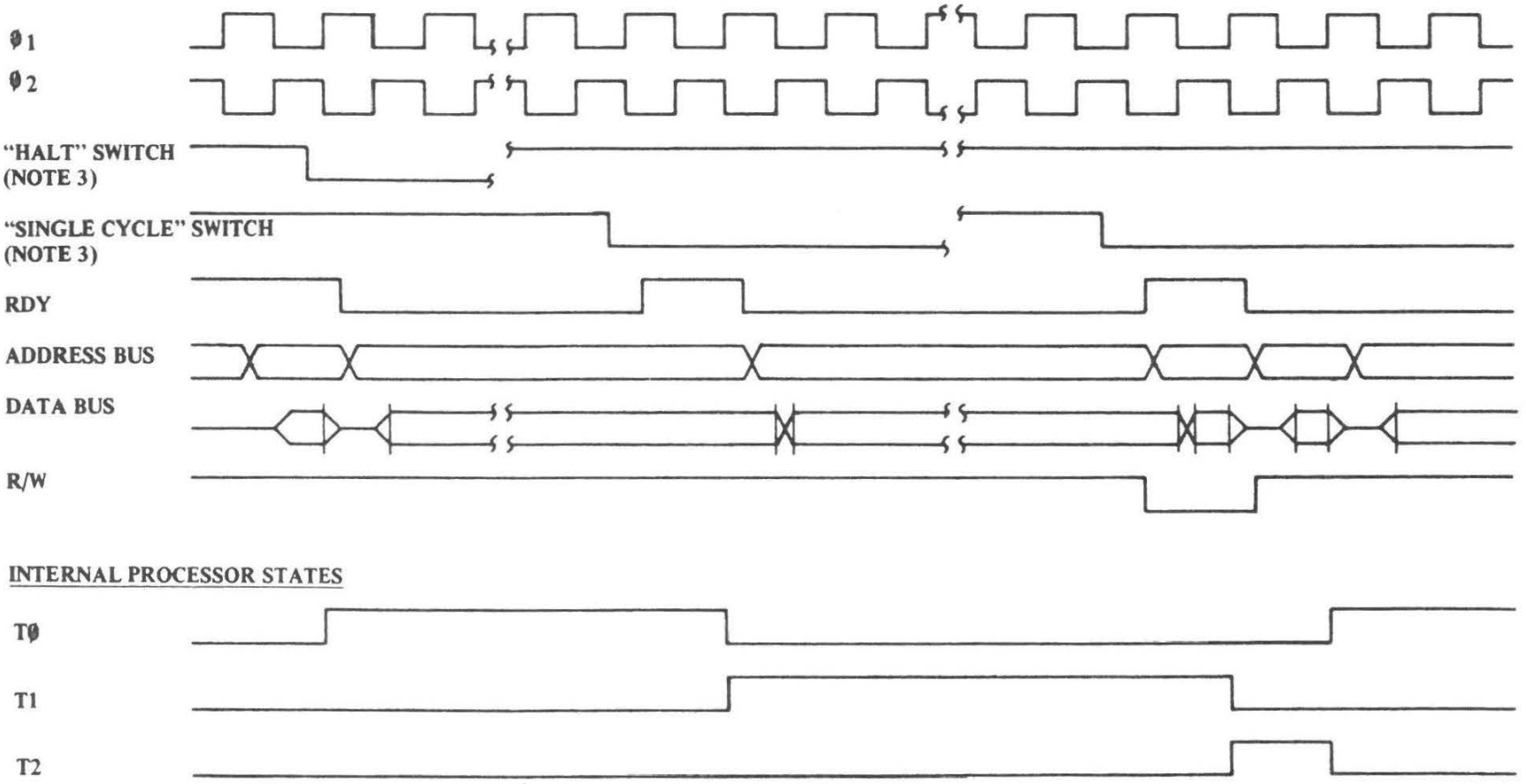
Im Einzelzyklusbetrieb kann der Prozessor in jedem Zyklus, außer dem Schreibzyklus, angehalten werden. So können alle Zyklen der Operation, in der Befehle geholt werden, genau untersucht werden. Das gilt



1	2-INPUT NAND	7400
2	"D" FLIP-FLOP	7474
3	2-INPUT OR	7432
4	INVERTER	7404
5	3-INPUT NAND	7410
6	2-INPUT NAND OPEN COLLECTOR	7403

Suggested Static Test Control Logic

ABBILDUNG 3.1: VORGESCHLAGENE STEUER-LOGIK FÜR STATISCHE PRÜFUNG



NOTES:

1. *INDICATES AN UNDETERMINED TIME PERIOD DURING WHICH THE SIGNAL WILL CHANGE.*
2. *THE DATA BUS ENTERS THE HIGH-IMPEDANCE STATE DURING EACH PHASE ONE PULSE. HOWEVER, WHILE THE PROCESSOR IS STOPPED THE DATA BUS WILL APPEAR TO REMAIN HIGH OR LOW AS SHOWN.*
3. *SWITCH ACTUATION IS INDICATED BY A LOW SIGNAL.*

Single Cycle Timing

ABBILDUNG 3.2: ZEITDIAGRAMM FÜR EINZELZYKLUS

auch für das Holen der Operanden. Auf diese Weise ist es möglich, die Arbeitsweise der meisten Hardware zu überprüfen, die in der Steuerung und Adressierung des Speichers beteiligt ist. Ebenso kann die Arbeitsweise der meisten peripheren Schnittstellen überprüft werden. Damit wird die benötigte Zeit zur Prüfung der dynamischen Arbeitsweise des peripheren Bausteins verkleinert. Wenn durch Betätigung des Einzelzyklusschalters der Prozessor in einen Schreibzyklus weiterschaltet, beendet er diesen Zyklus und hält im ersten Lesezyklus (R/W = 1) an, der folgt. Bild 3.2. zeigt das Zeit-Diagramm.

Anhang A enthält eine genaue Zusammenstellung der Daten, die während jedes Zyklus eines MCS 6502 Befehls auf den Adress- und Datenleitungen erscheinen müssen.

Der Prozessor gibt häufig eine Adresse aus und holt Daten, die er nicht verwendet. Dies ist eine Eigenschaft eines Prozessors, der das „look ahead" Verfahren zum Pipelining benutzt. Die Untersuchung des Signales SYNC ermöglicht die genaue Erkennung ob Daten, die aus dem Speicher geholt werden, innerhalb des Prozessors benutzt werden oder nicht. In Prototypsysteme kann eine sehr einfache „Datenfalle" eingebaut werden. Sie dient zur Untersuchung der Adresse und der Daten, die der Prozessor in WRITE-Zyklen erzeugt. Diese Falle kann den Inhalt des Adress- und des Datenbus speichern oder nur den des Adressbus. Das Letztere kann ausreichen, wenn eine getrennte Möglichkeit besteht, Daten im Speicher zu untersuchen (siehe Abschnitt 3.3). In Bild 3.3 wird eine vorgeschlagene Schaltung für die „Datenfalle" gezeigt. Diese Schaltung kann den Inhalt von Adress- und Datenbus für READ und WRITE Zyklen anzeigen. WRITE Daten werden gespeichert und während des nächsten READ Zyklus gehalten. Betätigen des Schalters Latch Reset öffnet die Eingänge zum Speicherregister und erlaubt die Überwachung der folgenden READ Zyklen.

3.1.3. Ausführung eines einzelnen Befehls

Es ist sehr nützlich die Ausführung jedes Befehls im einzelnen analysieren zu können. Oft genügt es jedoch den allgemeinen Programmfluß anzusehen. Besonders nützlich ist dies, wenn man Verzweigungen und Sprünge im Programm untersucht. Der MCS 6502 gibt ein SYNC Signal aus, das diese Möglichkeit bietet.

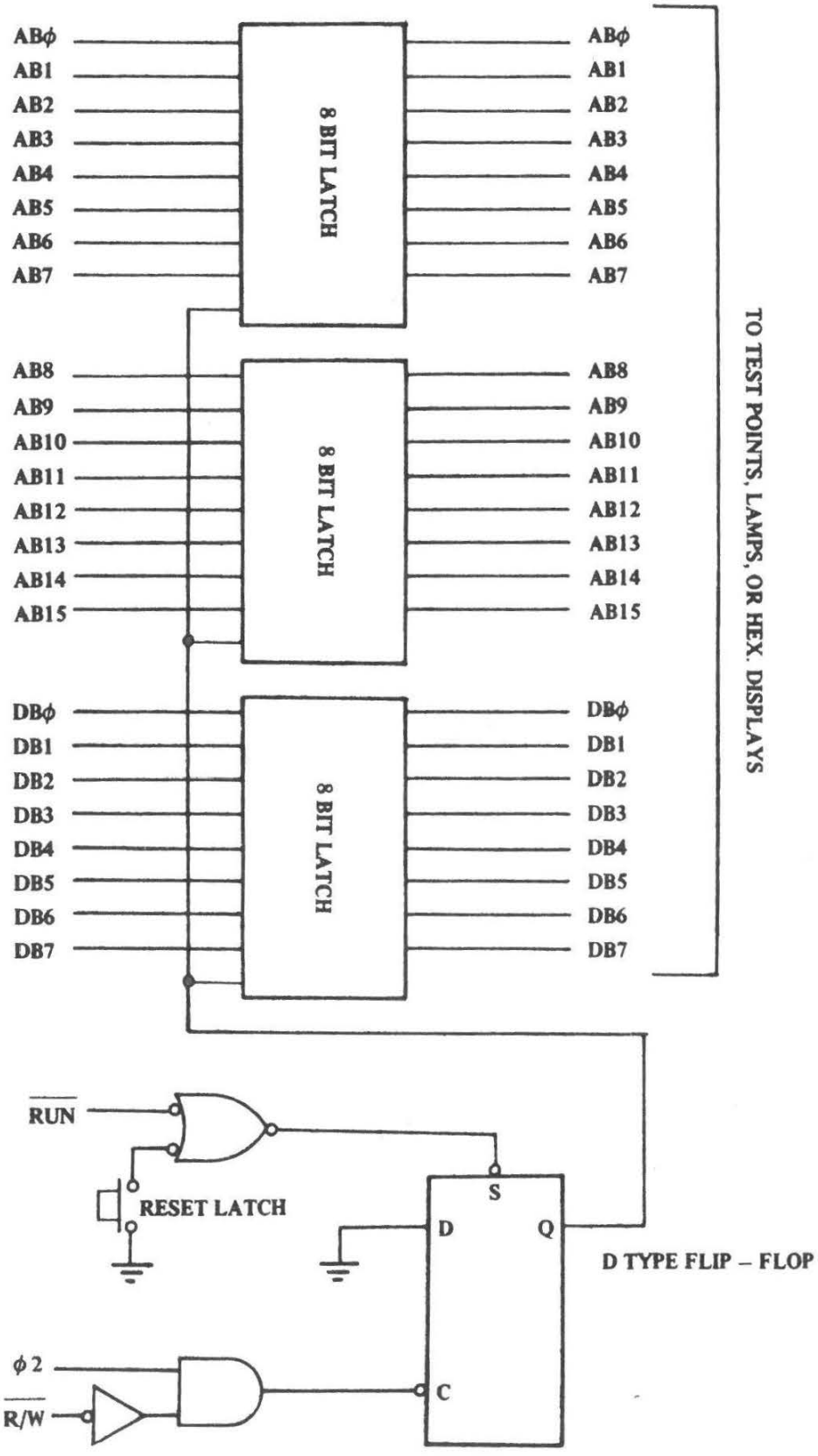
Die Arbeitsweise der Logik zur Ausführung eines Einzelbefehls beruht auf der Erzeugung des SYNC Signals innerhalb des Prozessors. Während jedes OP CODE Holzyklus auf H. Zur Ausführung eines Einzelbefehls wird SYNC benutzt um RDY auf L zu bringen. Unter diesen Voraussetzungen hält der Prozessor immer mit einer OP CODE Adresse auf dem Adressbus und dem OP CODE auf dem Datenbus an. Bild 3.4 zeigt das Zeitdiagramm dieses Vorgangs. Im Zeitdiagramm ist angenommen, daß der Prozessor im OP CODE Hol-zyklus angehalten ist. Betätigung des Schalters Einzelbefehl (single instruction) in Bild 3.1. löst Ausführung dieses Befehls aus. Der Prozessor hält dann an, wenn der nächste OP CODE geholt ist.

3.2 DYNAMISCHE PROFUNG

3.2.1 Einführung

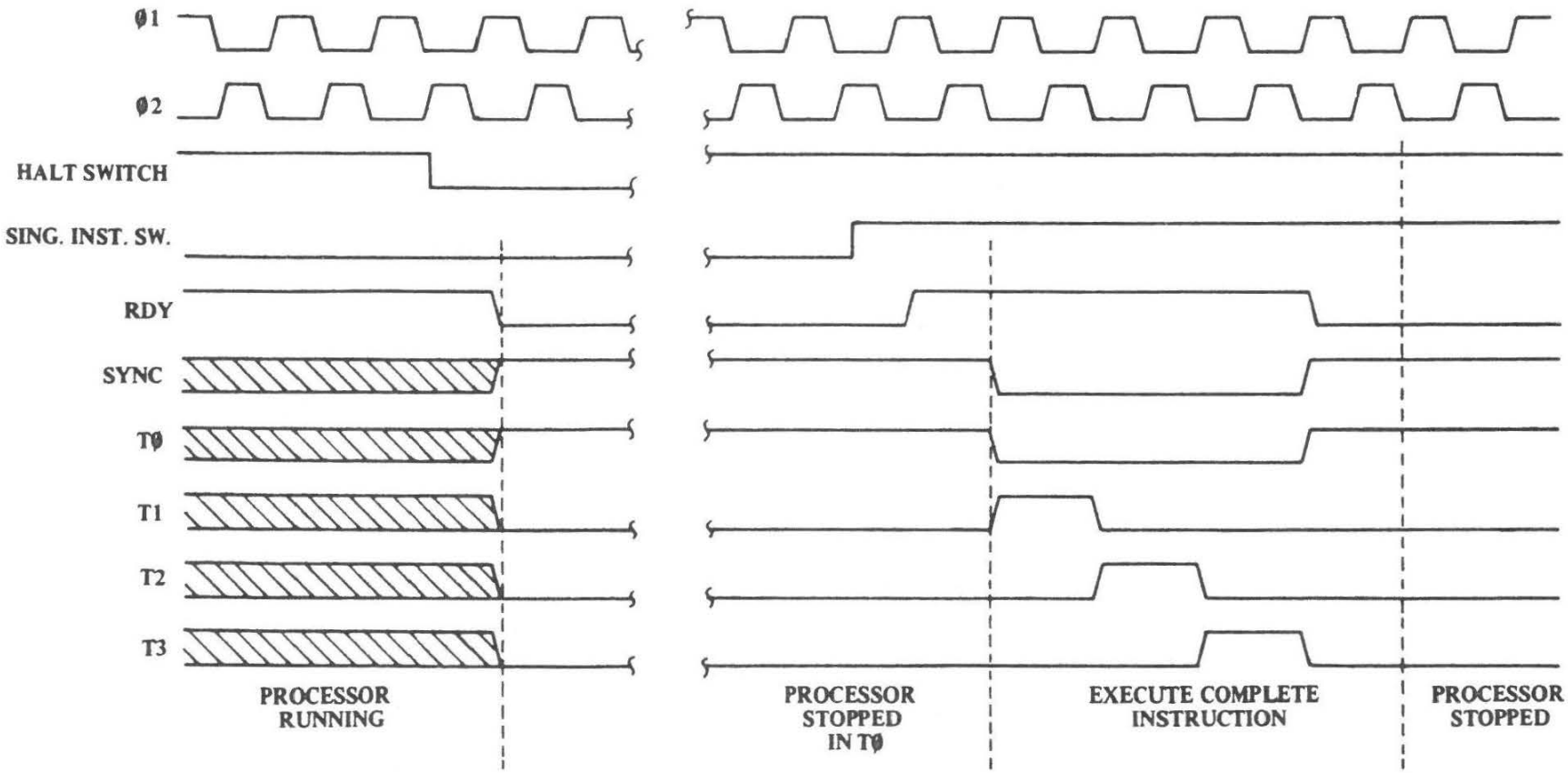
Mit statischen Prüfmethoden läßt sich die Arbeitsweise des größten Teils der Schnittstellen Hardware überprüfen, wie z.B. der Busexpander und Adressdekoder (zur Auswahl von ROM, RAM, usw.). Das ist jedoch nur der erste Schritt um einer richtigen Arbeitsweise des Systems sicher zu sein. Die meisten Peripherie-Bausteine können nur dann richtig geprüft werden, wenn der Prozessor mit voller Geschwindigkeit arbeitet. Das geht nicht ohne dynamische Prüfung.

Bei der dynamischen Prüfung führt der Prozessor im allgemeinen eine Programmschleife aus, d.h. er führt eine sich wiederholende Folge von Befehlen aus. Dadurch ist die Benutzung eines Oszilloskops



Microprocessor Single Cycle Data Trap

ABBILDUNG 3.3: DATENFALLE FÜR EINZELZYKLUS



Single Instruction Execution

ABBILDUNG 3.4: AUSFÜHRUNG EINES EINZELBEFEHLS

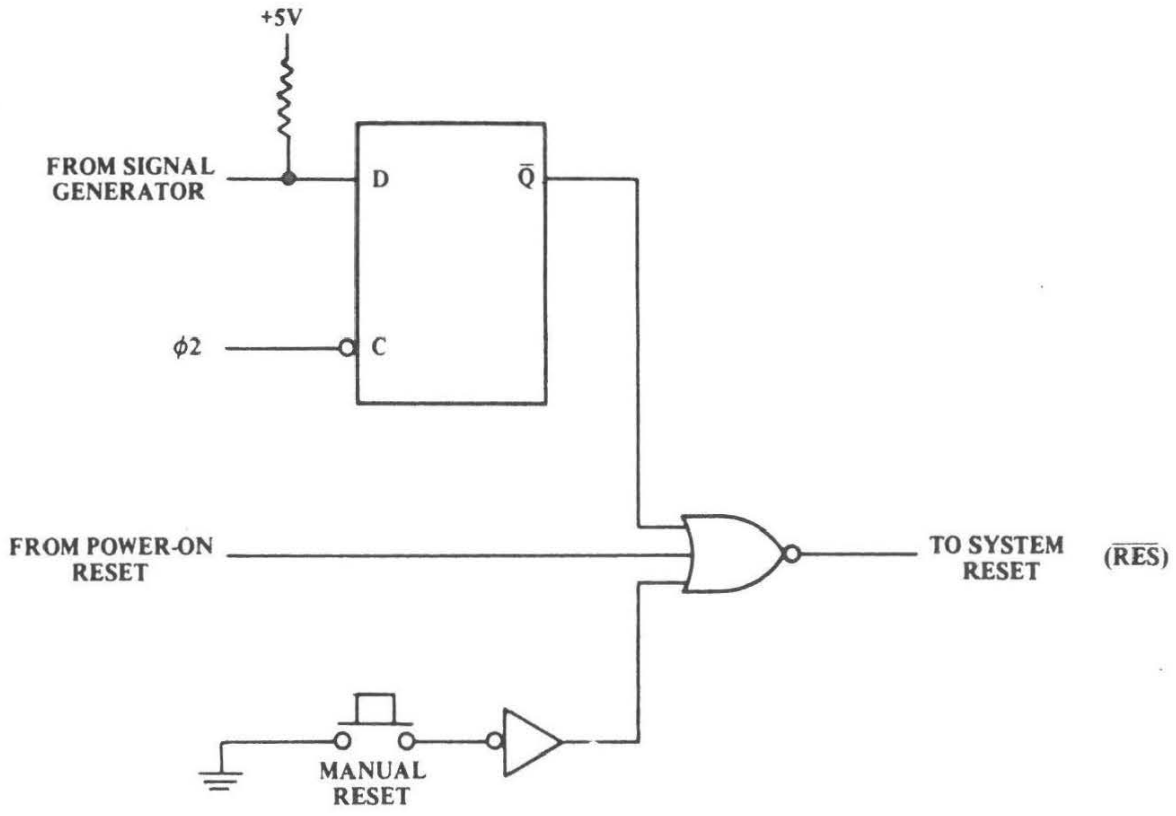
zur Überwachung des Prozessors möglich. Diese wiederkehrende Arbeitsweise kann von außen durch die Leitungen $\overline{RES}$ oder Interrupt ($\overline{IRQ}$ oder $\overline{NMI}$) hervorgerufen werden, oder sie kann Teil eines ausgeführten Programms sein. Beide Verfahren spielen eine wichtige Rolle bei der Überprüfung eines Systems.

3.2.2 Extern hervorgerufene Schleifen

Das direkteste Verfahren, den Prozessor zur Ausführung einer Schleife zu bringen, ist es einen der direkten Eingänge ($\overline{RES}$, $\overline{IRQ}$ oder $\overline{NMI}$) mit einem Signalgenerator zu treiben. Dieses Verfahren kann zur Fehlersuche in Systemen verwendet werden, die nur zum Teil arbeiten. Es erfordert nicht die richtige Ausführung einer bestimmten Menge von Befehlen, um eine Schleife hervorzurufen. Dies Verfahren kann nur benutzt werden, wenn ein Oszilloskop zur Überprüfung der Systemarbeitsweise verwendet wird. Dazu ist ein brauchbares Triggersignal nötig. Deswegen bespricht der folgende Abschnitt nicht nur die zu untersuchenden Signale und die Kurvenformen, die zu sehen sein sollten, sondern auch die Verfahren zur Erzeugung eines brauchbaren Triggersignals.

Die einfachste im Prozessor durchgeführte Operation ist der RESET. Wenn die RESET Hard- und Software nicht richtig arbeiten, kommt das System nie in seinen normalen Arbeitszustand. Aus diesem Grund wird als erste Funktion der $\overline{RES}$ - Eingang statisch und dynamisch geprüft.

Bild 3.5 zeigt eine vorgeschlagene Schaltung zur dynamischen Prüfung des RESET Einganges. In diesem Schaltbild wird der RESET - Eingang von einem Signalgenerator getrieben. Zwischen Signalgenerator und Prozessor befindet sich ein D-Flip-Flop um das RESET-Signal mit dem Prozessortakt zu synchronisieren. Diese Synchronisation ist sehr wichtig, um die Daten auf dem Oszilloskop stabil darzustellen.



Suggested Configuration
For Dynamic Reset Testing.

ABBILDUNG 3.5: VORGESCHLAGENE KONFIGURATION FÜR TEST DES DYNAMISCHEN RESET

Am wirkungsvollsten ist es zur dynamischen Prüfung das System am Anfang mit etwa einem Fünftel der Taktfrequenz zurückzusetzen. Damit kann der Prozessor die ersten Befehle der RESET-Folge ausführen, bevor er wieder anfängt. Dadurch ist eine genaue Untersuchung des Zeitablaufs von Adresse, Daten und R/B Signalen möglich. Mit der verzögerten Zeitablenkung, über die die meisten modernen Oszilloskope verfügen, kann jeder Teil des RESET-Vorganges untersucht werden.

Nachdem die richtige Arbeitsweise des RESET-Einganges überprüft ist, kann dasselbe Verfahren für den $\overline{IRQ}$ und den $\overline{NMI}$ - Eingang angewandt werden. Wenn der eine oder der andere dieser Eingänge mit einem Signalgenerator getrieben wird, der mit dem Prozessortakt synchronisiert ist, kann die Interrupt-Abruffolge genau untersucht werden. Dadurch ist ein Blick auf die Auswahllogik für periphere Schnittstellen möglich, um sicherzustellen daß alle Peripheriebausteine auf die richtige Adresse reagieren.

3.2.3. Software - Schleifen

Während der Oberprüfung des Systems muß der Entwickler die Arbeitsweise vieler einfacher Funktionen feststellen, die alle richtig arbeiten müssen, bevor das gesamte System arbeitet. Die Verwendung einfacher Software-Schleifen erlaubt eine genaue Untersuchung einer Funktion zur Zeit. Am wichtigsten ist, daß ein Oszilloskop verwendet werden kann, um Ereignisse zu untersuchen, die sehr selten vorkommen und normalerweise schwer zu erkennen sind.

Zur Ausführung einer Software-Schleife muß ein Programm geschrieben werden, daß mit einem JMP endet zum Anfang des Programms. Wenn der Prozessor einmal in diese Schleife eintritt, wird er die gleiche Folge von Befehlen solange ausführen, bis der RESET-Schalter betätigt wird.

Um eine Software-Schleife wirkungsvoll zu benutzen, muß es ein Ereignis geben das nur einmal pro Schleife geschieht. Dieses Signal wird zum Triggern des Oszillokopes verwendet. Bei einer einzelnen WRITE-Operation im Programm kann das R/W-Signal als Trigger benutzt werden. Durch sorgfältige Auswahl der Adressen im Programm kann eine Adressleitung als Trigger benutzt werden. Wenn kein anderes Signal verfügbar ist, kann das Setzen und Rücksetzen einer Leitung eines peripheren Schnittstellenbausteins zu Anfang des Programms ein sehr zweckmäßiges Triggersignal ergeben.

3.3 SYSTEMDIAGNOSE MIT HARDWARE HILFSMITTELN FÜR DEN PROGRAMMIERER

Zusätzlich zu den beschriebenen Verfahren in denen der Benutzer Oszilloskope und seinen Einfallsreichtum benutzt um Daten zu untersuchen, bietet MOS Technology, INC. verschiedene Hardware Hilfsmittel an, die bei der Fehlersuche in einem Mikrocomputer System helfen sollen. Weiter wird ein Software Hilfsmittel angeboten, der Emulator. Die Hardware Hilfsmittel sind ein Monitor für Tastatureingang (KIM, Keyboard Input Monitor) und ein Monitor für Fernschreibeingang (TIM, Teletype Input Monitor). Diese Hilfsmittel sollen die Fehlersuche in Mikroprozessor Programmen ohne die Hilfe von Oszilloskopen oder anderen Verfahren zum Festhalten von Daten dienen. Damit wird das Problem der Fehlersuche auf dieselben Verfahren verkleinert, die auf großen Computer Systemen zur Verfügung stehen.

Für alle diese Hilfsmittel, ob Hardware oder Software, wird vorausgesetzt, daß das Mikroprozessor System richtig verdrahtet ist und alle elektrischen Werte überprüft und für richtig befunden sind. Dann verbleibt als einziges Problem die Fehlersuche in Programmen und in der Eingangs/Ausgangs Hardware.

Die Hardware Verfahren gehen davon aus, das der Benutzer zu Anfang die Programmierung in irgendeiner Art Schreib/Lese Speicher vornimmt, der von einem Eingangs/Ausgangs-Gerät zu laden, zu untersuchen und zu ändern ist. Bei der Entwicklung ist dies der normale erste Schritt. Er sollte vor der Festlegung von Hardware in PROM oder änderbarem Speicher geschehen. Die einzige Ausnahme davon geht, wenn der Software Emulator benutzt wird und das Programm mit dem Emulator genau genug überprüft werden kann. In diesem Fall kann die Benutzung der Geräte KIM oder TIM bei der abschließenden Fehlersuche den Zugriff zu verschiedenen Speicherstellen ermöglichen. Das gemeinsame dieser Wege ist es, daß das Programm in einen Speicher eingelesen wird: in Falle des Emulators durch Lesen einer Eingangsliste, beim TIM durch Lesen Eines Eingangsbandes vom Ausgang des Cross-Assemblers und ein KIM durch Laden des Programms über die Tastatur.

Mit jedem dieser Verfahren kann der Benutzer die Anfangsbedingungen an verschiedene Speicher- und Registerstellen setzen und die Ausführung seines Programmes bei einer Speicherstelle beginnen. Es werden Verfahren verwendet, die das Anhalten des Programms an einer bestimmten Stelle erlauben, um die Ergebnisse der gerade abgeschlossenen Operation zu untersuchen. Wenn die Ergebnisse richtig sind, ist auch die Programmierung zwischen dem Anfangspunkt und dem Endpunkt richtig. Bei fehlerhafter Programmierung untersucht der Benutzer die Daten mit einem Eingangs/Ausgangs Gerät und der damit verbundenen Hardware oder Software, und bestimmt durch Durchsicht der Daten und Untersuchung seiner Programmierung, welcher Fehler die angezeigten Ergebnisse verursachen konnte.

Sollte die Untersuchung des Programms nicht ausreichen, hat jedes diese Systeme die Möglichkeit dasselbe Programm mit anderen oder den ursprünglichen Daten auszuführen. Dabei wird an früheren Haltepunkten angehalten, bis der Vorgang, der die falschen Daten erscheinen läßt, gefunden ist.

Der normale Ablauf eines Entwurfs sollte eine Kombination der verschiedenen Verfahren enthalten. Der Emulator ermöglicht eine einfache Untersuchung der Datenwege, Schleifen und der Leistung des Programms ohne die Benutzung von Hardware. Er ist besonders nützlich für den Benutzer, der Routinen entwickelt, die einen großen Teil an Schleifen oder Subroutinen oder einen fertigen Algorithmus enthalten.

Die Benutzung des Emulators hat folgende Vorteile:

1. Durch die Leistung einer großen Maschine wird die Verfolgung von Operationen ermöglicht, die auf der Hardware Ebene fast nicht möglich ist.
2. Es kann vor dem Zeitpunkt, wo die Hardware festgelegt wird, festgestellt werden ob mehr Speicher oder mehr Zeit zur Ausführung einer Operation nötig ist. Das kann zu einer großen Änderung der Hardware Realisierung führen.

Es ist auf jeden Fall keine sehr wirkungsvolle Verwendung der Zeit des Entwicklers ein Mikroprozessor System in Betrieb zu nehmen, ohne Assembler oder eine Schnittstelle, wie das TIM zu benutzen.

Für den Benutzer, der gerade auf dem Gebiet der Mikroprozessoren anfängt ist das Verfahren KIM annehmbar, da Länge und Kompliziertheit der zu schreibenden Programme nicht so groß sind. In diesem Falle können Programme direkt in Hex eingegeben und Fehler nur mit dem KIM gesucht werden.

3.3.1

Mit KIM können hexadezimale Werte in vorgegebene Speicherstellen eingegeben und Ergebnisse überwacht werden. KIM steht dem Systementwickler in verschiedenen Formen zur Verfügung. In seiner einfachsten Form kann ein einzelner Baustein des Typs MCS 6530 mit 124 Byte vorprogrammierten ROM in ein bestehendes System eingebaut werden. Der KIM Baustein enthält ein Monitorprogramm mit folgenden Eigenschaften:

a) Steuerung des Dateneingangs und -ausgangs für serielle Fernschreiber (ASR 33, Silent 700 usw.)

b) Dateneingang und Systemsteuerung von einer Tastatur mit 22 Tasten

c) Darstellung von Adresse und Daten auf eine 7 Segment-Anzeige mit 6 Stellen.

In einem Mikroprozessor System mit dem KIM Baustein können folgende Operationen ausgeführt werden:

a) Für Eingabe, Anzeige und Steuerung können Betriebsart Tastatur (KB) oder Fernschreiber (TTY) gewählt werden.

b) In der Betriebsart KB können Adress- oder Datenfelder von der Tastatur eingegeben werden. Der Inhalt jeder adressierbaren Stelle im System kann dargestellt und geändert werden (außer vorprogrammierten ROM Stellen). Mit der Taste STEP können aufeinanderfolgende Speicherstellen auf einfache Weise angezeigt werden. Mit der Taste RUN kann die Ausführung des Programms von jeder ausgewählten Startadresse erlaubt werden.

c) In der Betriebsart TTY können die Daten in jeder Speicherstelle ausgedruckt werden. Ebenso können Daten in jeder Speicherstelle geändert werden. Von jeder Startadresse zu jeder Endadresse kann ein Programmausdruck angefordert werden. Papierbänder können geladen oder automatisch erzeugt werden. Außerdem kann die Programmausführung von jeder ausgewählten Startadresse eingeleitet werden.

d) In beiden Betriebsarten beendet der Benutzer die Programmausführung durch die Taste STOP, die die Steuerung des Systems an das TIM Programm zurück gibt. Die Betätigung der Taste RST verursacht eine gesamte Rücksetzung des Systems und die Übergabe der Systemsteuerung an das KIM Programm

Der KIM Baustein steht auch als Teil eines Subsystems zur Verfügung. Diese gedruckte Leiterplatte enthält folgende Bauteile:

a) MCS 6502 Mikroprozessor Baustein
b) MCS 6530 - 002 Baustein (enthält KIM Monitor-Programm)
c) Tastatur mit 22 Tasten und Betriebsart Wahlschalter

d) 7-Segment LED Anzeige mit 6 Stellen

e) 1024 x 8 RAN

f) MCS 6530 - 003 Baustein mit Intervall-Timer, 16 Eingangs/Ausgangsstiften und 64 Byte RAN

g) Alle Schnittstellen Schaltkreise für die Verbindung mit seriellen Fernschreibern.

Dieses Subsystem bietet dieselben Eigenschaften wie oben beschrieben. Es wird aber als betriebsfertige Einheit geliefert, die nur an die +5 Volt Stromversorgung angeschlossen werden muß, um arbeiten zu können. Bei Verwendung als alleinstehendes Subsystem können Programme bis zu 1024 Schritten eingegeben und fehlerfrei gemacht werden und die Wirkungsweise von bis zu 16 Eingangs/Ausgangsstiften gesteuert werden.

Für weitere Einzelheiten über die mechanischen Abmessungen und die Arbeitsweise des KIM Bausteins und Subsystems wird der Leser auf das KIM Handbuch verwiesen, das getrennt geliefert wird.

3.3.2 TIM - Monitor für Fernschreibeingang

TIM ist ein vorprogrammiertes MCS 6530. Der Monitor für Fernschreibeingang erlaubt die Verwendung eines ASCII - Gerätes (Fernschreiber, Datensichtgerät) mit dem seriellen ASCII - Verfahren zur Verbindung mit dem Mikroprozessor. Auf diese Weise kann der Benutzer den Speicher von der Tastatur, vom Lochstreifen oder Kasetten laden, die er an sein Gerät angeschlossen hat. Durch Hinzufügung eines einzigen TTL - Bausteins zum System kann TIM als Anfangspunkt für den Mikroprozessor eingesetzt werden. Sobald die Anfangsbedingungen hergestellt worden sind, überträgt er sich selbst aus dem Anfangsspeicher heraus, ändert den Rest des Mikroprozessor Speichers in seine normale Konfiguration und arbeitet für den Mikroprozessor transparent.

TIM wird hauptsächlich verwendet nachdem festgestellt worden ist, daß das System richtig verdrahtet ist. Dazu werden die bereits beschriebenen Verfahren verwendet. Zur Fehlersuche können dann vorgegebene Werte eingegeben werden, Teile des Programms ausgeführt und die Ergebnisse untersucht werden.

Da die Eingangs/Ausgangs Bausteine eine Erweiterung des Speichers sind, ist die Fehlersuche vereinfacht. Fehlersuchverfahren können Eingangs/Ausgangs-Geräte steuern, um Lampen anzuschalten, Schalter zu überwachen, Motoren zu starten und stoppen usw. Z.B. können alle Verbindungen zu Lampen und Schaltern von der Fernschreibertastatur überprüft werden. Dazu wird der entsprechende Code in die Eingangs/Ausgangs-Register eingeschrieben, der die Lampen anschaltet. Richtige Arbeitsweise der Schalter kann ohne laufendes Programm überprüft werden. Die Schalter werden dazu in jede ihrer Stellungen gebracht und über den Eingangs/Ausgangs-Baustein wird das Ergebnis dem Programmierer angezeigt.Mit dieser Art Überprüfung werden alle Eingangs/Ausgangs-Verbindungen untersucht, um sicher zu gehen, daß der Eingangs/Ausgangs Baustein an der richtigen Speicheradresse sitzt, die Verdrahtung einwandfrei ist und überprüft den Mikroprozessorbus.

Eine vernünftige Verfahrensweise zur Benutzung vom TIM oder KIM ist die Verbindung des Baustein mit dem System um den Mikroprozessor durch die Ein-Schritt-Anfangssequenz zu bringen. Anschließend wird die Fehlersuchmöglichkeit des TIM benutzt alle Eingangs/Ausgangs Verbindungen auf Richtigkeit zu überprüfen, bevor das Benutzerprogramm ausgeführt wird. Durch Abklemmen oder Ziehen von Steckern kann in Fällen wie dem Bremsen von Motoren oder anderen Geräten, die eine bestimmte Zeitfolge erfordern die richtige Verbindung überprüft werden ohne daß das Gerät tatsächlich angeschlossen ist. In diesem Fall wird ein Oszilloskop oder ein Meßinstrument benutzt, um festzustellen, daß der Einfluß des Mikrosprozessors an der Meßstelle wie erwartet ist. Dies Verfahren wird als zweiter Schritt der Inbetriebnahme empfohlen.

Im Abschnitt über die Benutzung der Wiederstart oder Startfolge und des Einzelzyklusbetrieb werden wichtige Einzelheiten zur Oberwachung der Verbindungen des größten Teils des Systems angegeben. Diese Anweisungen waren unanhängig von der Art der Programmierung, die dem Programmierer zur Verfügung stand.

Der Vorteil der Verwendung von TIM oder KIM bei der Anfangsüberprüfung ist es, daß es ein bekanntes Programm gibt, das mit Sicherheit richtig ist und während dieser Anlauffolge abgerufen werden sollte. In der Dokumentation für TIM oder KIM kann der Benutzer die Kodierung des ROM nachsehen und diese bekannten Folgen aus dem TIM oder KIM Programm benutzen, um seine Anfangsfolge zu überprüfen. Dadurch wird eine weitere Veränderliche ausgeschaltet. Deshalb sollte die anfängliche Systemüberprüfung zunächst mit TIM oder KIM Programm in der Anfangsfolge geschehen, um sicher zu gehen, daß die Verbindung zum TIM und zum Speicher in Ordnung sind. Nachdem einmal die grundlegende Arbeitsweise des TIM überprüft ist, gibt es eine bekannte Folge durch die der TIM dynamisch gehen wird, der dem Benutzer erlaubt die richtige Arbeitsweise des TIM festzustellen. Anschließend wird der Rest der Verbindungen zum Speicher und zum Ein/Ausgang durch Schreiben und Lesen in Speicherstellen überprüft. Dazu wird die Fehlersucheigenschaft des TIM oder KIM benutzt. So werden Verbindung und Arbeitsweise jedes Bausteins des Systems überprüft ebenso alle Verbindungen mit außenliegenden Bausteinen.

Jetzt muß festgestellt werden ob das Programm in Ordnung ist. Das Benutzerprogramm kann entweder über die Tastatur oder über Hilfsgeräte geladen werden.

Die Fehlersuche kann auf das Programm beschränkt bleiben. Es ist nicht nötig, sich Gedanken darüber zu machen, ob es ein Hardware oder Software Problem ist. Durch Definition ist das Problem auf Programmierfehler reduziert, mit Ausnahme falscher Zeitabläufe für Eingangs/Ausgangs Bausteine.

Der Benutzer wird für eine genaue Besprechung der Programmierung mit TIM auf das getrennt gelieferte TIM Handbuch hingewiesen.

3.4 VERFAHREN ZUR INBETRIEBNAHME DES MIKROPROZESSORS

3.4.1. Einführung

Dieser Abschnitt soll alle vorher besprochenen Arbeitsweisen zu einem geordneten Verfahren zusammenfassen. Dies Verfahren beruht auf der Erfahrung, die bei der Inbetriebnahme von Systemen mit Prozessoren verschiedener Hersteller gewonnen wurde. Obwohl mit Sicherheit kein einzelnes Verfahren alle Software und Hardware Fehler in einem Mikrocomputer-System entdecken kann, sollte dieses Schritt für Schritt Vorgehen dem Entwickler erlauben, sein System mit der kleinstmöglichen Schwierigkeit in Betrieb zu nehmen.

Für dies Verfahren wird angenommen, daß eine Steuerlogik für Einzelzyklus und/oder Einzelbefehl vorhanden ist. Jedes der Hilfsmittel zur Systementwicklung, in Abschnitt 3.3. besprochen, hilft dem Benutzer bei der Inbetriebnahme seines Systems. Diese Geräte erlauben eine einfache Eingabe von Prüfprogrammen und die Änderung des Systemprogramms und der Daten.

Jeder Schritt im Verfahren enthält folgende Information:

- Zu prüfende Detail der System Hardware oder Software.
- Hardware, Prüfgeräte, usw. zur Ausführung der Prüfung.
- Maßnahmen zur Durchführung der Prüfung.
- Erwartete Ergebnisse
- Vorgeschlagene Verfahren zur Untersuchung von Fehlerzuständen.

Es kann sehr leicht geschehen, daß mehrere Probleme sich gegenseitig so beeinflussen, daß innerhalb des Systems nichts richtig zu arbeiten scheint. Die Beseitigung jeweils eines Problems zur Zeit ergibt schließlich ein vollständiges, arbeitendes System mit einem Minimum an Frustration.

3.4.2 Stromversorgung des Systems - Schritt 1

Es wird empfohlen, die ersten Prototypen von Mikrocomputer-Systemen unter Verwendung von Fassungen von IC (Prozessor, Speicher usw.) aufzubauen. Ein klarer Vorteil dieses Verfahrens ist die Möglichkeit zur Überprüfung von V_{DD} und V_{SS} an jeder Fassung vor dem Einstecken der Bausteine. Die Spannung zwischen der V_{DD} - Leitung sollte innerhalb der Toleranzen um 5 Volt zur Leitung V_{SS} liegen. Mit diesem grundlegenden ersten Schritt können Stromversorgungsverbindungen vermieden werden, die den Bausteinen des Systems schaden können.

Nach der Überprüfung mit Voltmeter oder Oszilloskop wird der Prozessor in seinen Sockel gesteckt. Es wird überprüft, ob der zusätzliche Strom innerhalb der Daten für diesen Baustein liegt.

Vor dem Einstecken der anderen Bausteine werden die Adressleitungen, die SYNC - Leitung (6502) und die Ausgangstakte (6502, 6503,) untersucht, um sicher zu gehen, daß der Prozessor Signale erzeugt. Die Adressleitungen sollten aufwärts zählen und die SYNC - Leitung sollte regelmäßige positive Impulse erzeugen. Für diese Prüfung müssen die Leitungen $\overline{RES}$ und RDY auf H (+ 2,4 V) liegen.

Wenn der Prozessor zu arbeiten scheint und der Stromverbrauch innerhalb der erlaubten Grenzen liegt, können die restlichen Bausteine im System in ihre Fassungen gesteckt werden.

3.4.3. Grundlegende Zeitbeziehung im System - Schritt 2

Bevor ein Mikroprozessor System arbeiten kann, müssen die grundlegenden Zeitbeziehungen im System (Takt 1, Takt 2, usw.) überprüft werden. Das Bedeutendste dieser Signale ist der Systemtakt.

Es sind die Prozessor-Ausgangstakte zu untersuchen. Beim 6502 sind beide Phasen (Takt 1 und Takt 2) verfügbar um den Rest des Systems zu treiben. In diesem System muß die Zeitfolge des Taktes sehr genau überprüft werden, um sicher zu gehen, daß die Taktsignale im Prozessor mit den Taktsignalen für die Unterstützungsbausteine kompatibel sind.

Mit einem Oszilloskop werden der Eingangstakt 1 und der Ausgangstakt 2 für die Unterstützungsbausteine verglichen, um festzustellen, daß die Verzögerung durch den Taktpuffer nicht die vorgeschriebene Grenze überschreitet.

3.4.4 Rücksetzen des Systems - Schritt 3

Statische und dynamische Untersuchung der Funktion Reset ergibt eine genaue Information über die Wirkungsweise des Systems. Dieser Schritt überprüft die Arbeitsweise des größten Teils der System-Hardware.
Dazu sind nötig:

- Steuerlogik für Einzelzyklus/Einzelbefehl
- Oszillioskop
- Signalgenerator (zum Treiben des RESET)

3.4.4.1 Statische Untersuchung von Systemeinzelheiten

Betätigen Sie die Taste HALT und dann den Schalter für RESET von Hand. Dann wird der Schalter für den Einzelzyklus 6 mal gedrückt. Dadurch geht der Prozessor schrittweise durch den ersten Teil der BRK - Folge und zum Holen des RESET - Vektors. Zu dieser Zeit sollte der Prozessor auf dem Adressbus FFFC erzeugen und das ROM sollte das niederwertige Byte des RESET - Vektors als Antwort auf den Datenbus gelegt haben. Jetzt ist eine gute Gelegenheit die folgenden grundlegenden Einzelheiten zu überprüfen:

A. Adressleitungen:
Mit dem Oszilloskop werden die Logikpegel auf den Adressleitungen überprüft. Sie müssen auch über alle Bus-Expander zum Speicher und den peripheren Bausteinen übertragen werden. Diese Prüfung ist sehr wichtig, da ein falscher Anschluß der Bus-Expander ein häufig vorkommender Schaltungsfehler ist.

B. ROM/PROM - Auswahlleitungen:
Mit dem Oszilloskop wird überprüft, daß die Adresse FFFC das ROM auswählt, daß das niederwertige Byte des RESET - Vektors enthält.

C. Datenbus:
Mit dem Oszilloskop werden die Spannungen auf den Datenbus-Anschlüssen des Prozessors überprüft. Diese Signale müssen am Prozessor untersucht werden, um sicher zu stellen, daß Bus-Expander für zwei Richtungen im System richtig arbeiten. Das häufigste Anzeichen für eine falsche Arbeitsweise der Datenbus-Expander sind „schwebende" Stifte des Prozessor-Datenbus. D.h. die Stifte des Prozessor-Datenbus werden weder auf H oder L getrieben, da die Bus-Expander hochohmig oder umgekehrt sind.

D. Verschiedene Prozessor-Anschlüsse:
Mit dem Oszilloskop werden die anderen Prozessor-Anschlüsse (R/W, $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$, usw.) überprüft, ob die Pegel an diesen Punkten stimmen. Beide Interrupt -Eingänge und der R/W-Ausgang müssen auf H liegen. Überprüfen Sie das Signal R/W am Eingang des Speichers und der peripheren Bausteine.

Wenn diese Anfangsprüfungen fertig sind, kann durch einmaliges Betätigen des Schalters Einzelschritt das höherwertige Byte des Interrupt-Vektors von der Adresse FFFD geholt werden. Bei der nächsten Betätigung des Schalters Einzelschritt sollte der Adressbus des Prozessors den RESET-Sektor enthalten, der aus dem Speicher geholt wurde.

Jetzt kann der Prozessor die Routine zur Herstellung der Anfangsbedingungen ausführen. Es kann angenommen werden das während dieser Routine Programmspeicher angesprochen, Peripherie-Register geladen und Register im Prozessor gelöscht oder auf einen Anfangswert gesetzt werden. Es ist sehr zweckmäßig, diese Routine Befehl für Befehl auszuführen. Dabei wird festgestellt, daß bei jedem Zugriff zum Programmspeicher der richtige Befehl zurückkommt. Ohne eine Datenfalle ist es jedoch sinnvoller zur Untersuchung der Wirkungsweise der Peripherie-Bausteine dynamische Untersuchungsverfahren zu verwenden, da die meisten Zugriffe zur Peripherie zum Zweck des Einschreibens in die Eingangs/Ausgangs-Steuerung oder die Steuer-Register in den Peripherie-Bausteinen dienen.

3.4.4.2 Dynamische Untersuchung von System-Einzelheiten

Das allgemeine Verfahren der dynamischen Untersuchung ist in Abschnitt 3.2 besprochen worden. Die folgende Erläuterung benutzt dies Verfahren zur Untersuchung vieler Einzelheiten der Systemarbeitsweise.

Nehmen Sie das System wie in Abschnitt 3.2.2. beschrieben in Betrieb. Wenn die Prüfgeräte richtig arbeiten kann der größte Teil der Systemarbeitsweise nur mit dem Oszilloskop überprüft werden.

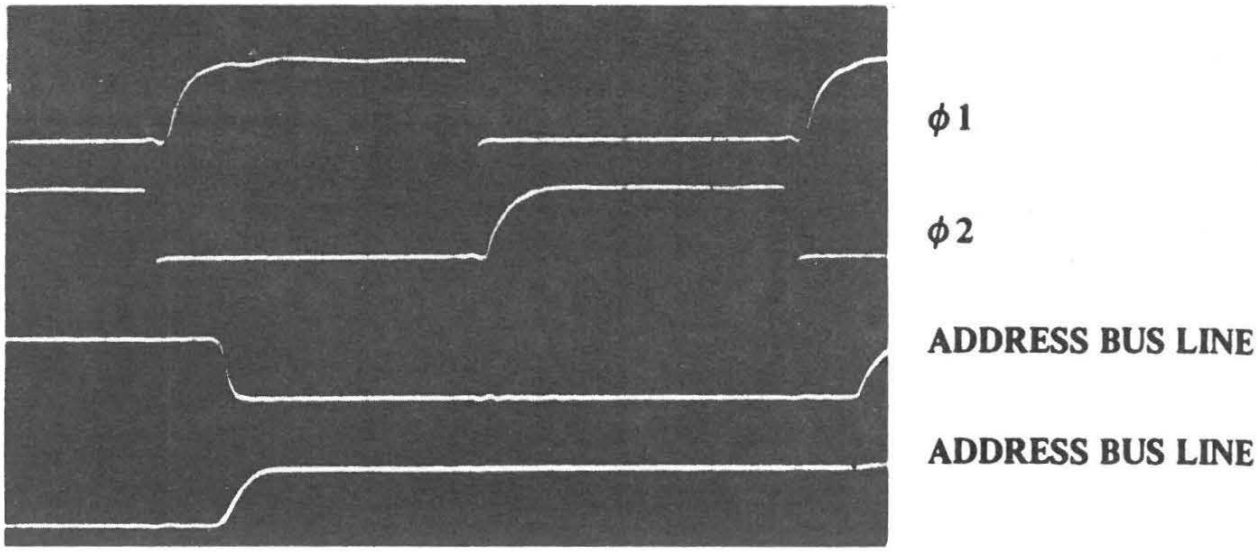


ABBILDUNG 3.7A: RICHTIGE ADRESSENLEITUNGEN

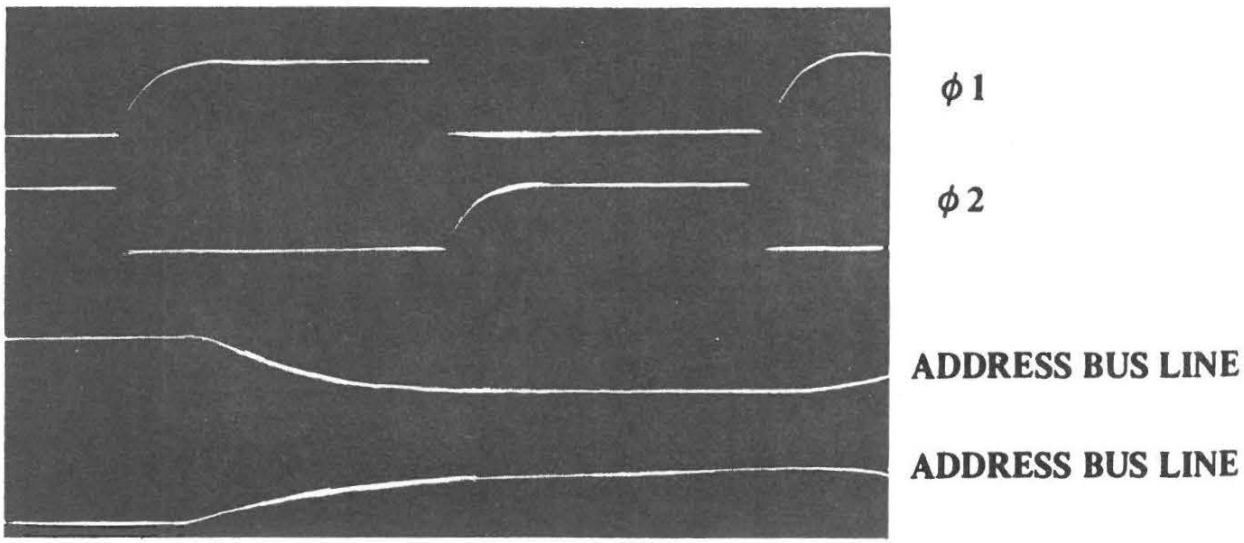
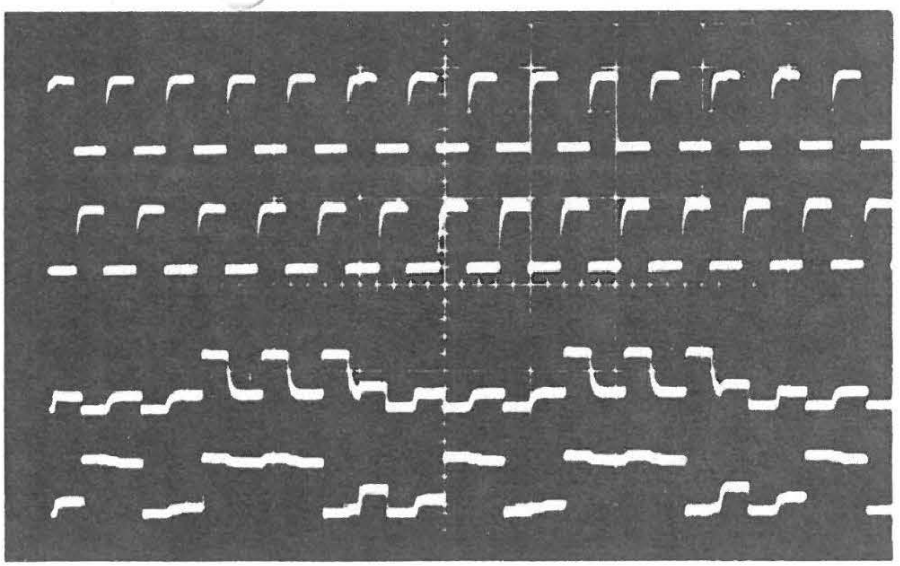
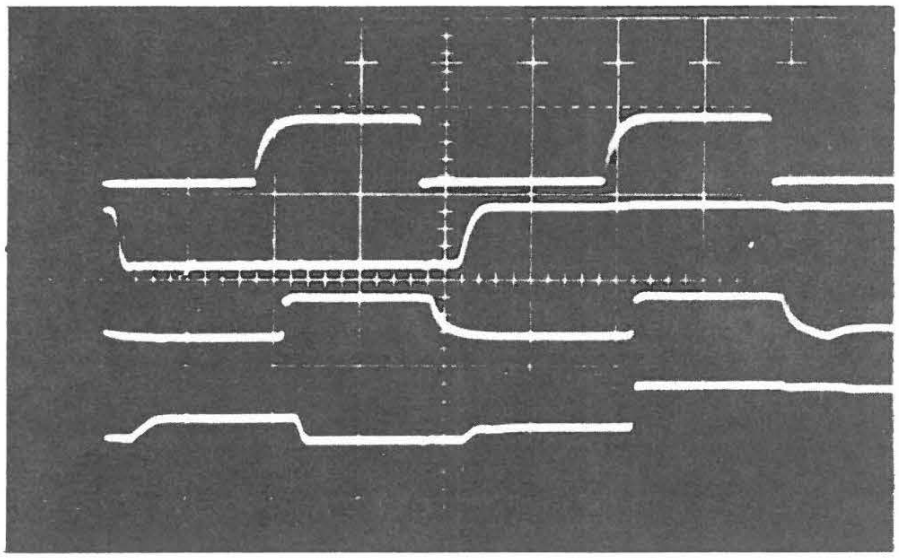


ABBILDUNG 3.7B: ADRESSENLEITUNGEN MIT ÜBERLAST

Address Lines In MCS650X Systems

ABBILDUNG 3.7: ADRESSLEITUNGEN IM MCS 650X SYSTEM





The Data Bus in MCS650X Systems

ABBILDUNG 3.8: DER DATENBUS IM MCS 650X SYSTEM

3.4.4.2.1 Oberprüfung des Adress-Bus

Als erstes wird die Zeitfolge auf den Adress-Leitungen untersucht. Diese Leitungen wechseln während des ersten Teils von Takt 1. Nach der festgelegten Zeit sollten Sie sich einschwingen und während des Restes des Zyklus stabil bleiben.Bild 3.7a zeigt die Kurvenform die bei Untersuchung von Takt 1, Takt 2 und zwei Adress-Leitungen erwartet werden kann. In diesem Bild geht eine Adress-Leitung auf H und die andere auf L. Diese Signale werden im Prozessor erzeugt und arbeiten mit Sicherheit richtig, solange die Belastung an den Stiften innerhalb der Daten liegt. Der häufigste Grund für Probleme mit Spannungspegel und Anstiegszeit ist Überlast. Ein zu hoher Pegel bei L zeigt normalerweise, daß der Adresspuffer einen zu hohen Strom treiben muß. Eine übergroße Lastkapazität bewirkt zu lange Anstiegs- und Abfallzeiten (Bild 3.7b).

Bei der Untersuchung von Adressleitung ist es wichtig, daß die Daten am Prozessor und direkt an den verschiedenen Unterstützungsbausteinen untersucht werden. Dadurch ist sichergestellt, daß die Bus-Expander im System richtig arbeiten und die Adressen dort stimmen, wo sie tatsächlich verwendet werden.

3.4.4.2.2 Oberprüfung des Datenbus

Nach Oberprüfung der Adressen folgt als nächster Schritt die Untersuchung des Datenbus. Damit wird die Gültigkeit der Daten, die vom Prozessor zu den Unterstützungs-Bausteinen und von den Unterstützungs-Bausteinen zum Prozessor zurück übertragen werden bestätigt.

Bild 3.8 veranschaulicht die Kurvenform, die man auf den Datenbus-Leitungen erwarten kann. Während Takt 1 ist die Spannung auf dem Datenbus nicht vorhersehbar, da weder der Prozessor noch die Unterstützungs-Bausteine diese Leitungen treiben.

Während Takt 2 sollten die Datenbusstifte aber auf H oder L gehen. Nur während Takt 2 (H) kann die Gültigkeit der Daten bestätigt werden.

Bei Untersuchungen des Datenbus sind drei wichtige Parameter zu beachten. Das sind Spannungspegel, die Zeit zu der die Daten gültig sind und die Verzögerung von der fallenden Flanke von Takt 2 bis zur Ungültigkeit der Daten.

A. Spannungspegel:
   Die Logikpegel auf dem Prozessor Datenbus müssen für eine logische 1 größer als 2,4 Volt und für eine logische 0 kleiner als 0,4 Volt sein. Das ist eine sehr grundlegende Voraussetzung, aber eine schnelle Überprüfung dieser Pegel zu Anfang der Überprüfung kann dem Entwickler viele Stunden sparen, in denen er versuchen würde, ein System zum Arbeiten zu bringen, dessen Signale oberflächlich als ausreichend erscheinen.

   Weiterhin ist es wichtig zu überprüfen, ob die Spannung für logisch 0 tatsächlich negativ (unter V_{BD}) geht. Die logischen Signale in alle Baustein-Eingänge dürfen nicht unter - 0,3 Volt wie in den Daten angegeben gehen.

B. Zeit für gültige Daten:
   Die Zeit zu der die Daten gültig werden, gibt die Gesamtzeit an, die Prozessor oder Speicher zur Verfügung stehen, um Data und Speicher zu stabilisieren, die im Baustein Daten festhalten. Aus diesem Grund dürfen die Daten nicht zu lange brauchen, um entweder ein gültiges H oder ein gültiges L zu erreichen. Der Hauptgrund für langsame Signale auf dem Datenbus ist eine übermäßige Belastung, entweder resistiv oder kapazitiv. Überprüfen Sie die mit dem Bus verbundenen Bausteine sorgfältig um sicher zu gehen, daß die Gesamtbelastung innerhalb der Daten liegt.

C. Haltezeit:
Die Haltezeit ist als die Zeit definiert zwischen der fallenden Flanke des Takt 2 Pulses und dem Zeitpunkt, an dem die Daten nicht länger gültig sind. Der Prozessor braucht zum Festhalten der Daten in seinen Eingangs Flip-Flops eine minimale Haltezeit von 10 ns. Der innere Takt 2 Impuls des Prozessors wird verwendet, um den Inhalt des Datenbus auf diese Speicher zu geben. Die verschiedenen Unterstützungs-Bausteine innerhalb des Systems brauchen auch eine Haltezeit. Überprüfen Sie die Signale sorgfältig wie Sie an den RAM, ROM, usw. erscheinen um festzustellen daß jedes innerhalb seiner betrieben wird.

3.4.5. Genaue Untersuchung von Einzelteilen

Nach der dynamischen Überprüfung der RESET - Routine wird versucht das System-Programm laufen zu lassen. Der Erfolg dieses Arbeitsganges bestimmt, ob es nötig ist eine weitere genaue Untersuchung der Bauteile vorzunehmen. Die Überprüfung des System-Programms sollte Schritt für Schritt vor sich gehen, genau wie die Hardware Überprüfung. Wenn alle Bausteine, Datenwege usw. im System sorgfältig untersucht sind, kann die Überprüfung der Software unter der Annahme vor sich gehen, daß die Hardware voll funktionsfähig ist. Es ist jedoch unvermeidlich, daß Zweifel aufkommen. Bei der Überprüfung der Hardware gibt es Zeiten, in denen das Programm falsch zu sein scheint. Daten gehen nicht in den Speicher, wie sie sollten, oder im allgemeinen wird ein Hardware-Fehler angezeigt. Sobald dies geschieht sollten die verdächtigen Bauteile im einzelnen untersucht werden. Mögliche Hardware-Probleme sollten die Überprüfung der Software nicht zunichte machen. Es ist dabei wichtig, sich an die Schritt für Schritt Methode zu halten.

Es gibt zwei Arten Bauteilprobleme: Bauteilfehler, d.h. ein Teil das nicht seinen Daten entsprechend arbeitet, oder ein Systemfehler, d.h. ein Bauteil das auf falsche Weise in dem System verwendet wird. Letzteres Problem kann das Ergebnis falschen Systementwurfs oder falscher Verdrahtung sein. Hier wird das Problem einwandfreier Bauteile angesprochen, die im System nicht vorschriftsmäßig arbeiten. Wenn es irgendeinen Zweifel über die Funktions-Fähigkeit eines Bauteils gibt, sollte es sofort dann ersetzt werden, wenn feststeht, daß es einwandfreie Signale an allen Eingängen bekommt. Wenn es dann noch nicht richtig arbeitet ist das Problem vermutlich systembezogen.

Die genaue Überprüfung eines Bauteils wird am wirkungsvollsten durch Laden eines kleinen Schleifenprogramms in das RAM des Systems durchgeführt. Aus diesem Grund können die Fehlersuch-Software von TIM oder KIM (siehe TIM und KIM Handbücher) von großem Wert sein. Das Verfahren enthält statischen und dynamischen Betrieb eines kleinen Prüfprogramms, das jedes Bauteil im System benutzt. Das Ziel dieses Schrittes sollte eine vollständige Bestätigung sein, daß alle Baustein-Auswahlleitungen, alle Daten-Adressleitungen technisch arbeiten und daß die Unterstützungs-Bausteine den Prozessor richtig treiben. Das vorgeschlagene Verfahren für die Überprüfung jeder Bauteilart wird getrennt unten besprochen.

A. ROM (PROM):

Das einfachste Bauteil in jedem Mirkoprozessor System ist das ROM. Dieser Baustein gibt als Antwort auf eine Adresse ein 8-bit Wort auf den Datenbus. Schwierigkeiten mit ROM werden üblicherweise durch falsche Baustein-Auswahlsignale oder durch falsche Anwendung von Bausteinen hervorgerufen, die nicht Teil der MCS 6500 Familie sind. Daher ist die statische Prüfung von ROM im allgemeinen ein sehr wirkungsvoller erster Schritt. Dazu muß ein Prüfprogramm in das RAM eingegeben werden und mit dem Schalter für Einzelzyklus ausgeführt werden.

Das Programm sollte einfach einen Lesebefehl (z.B. LDA oder LDX) eines ausgewählten Wortes auf jedem ROM Baustein, der geprüft werden soll, ausführen. So können die Baustein-Auswahlleitungen die Adressen für die Bausteine und die auf den Datenbus gelegten Daten untersucht werden.

Nachdem Baustein-Auswahl,Adressbus und Datenbus statisch überprüft sind, kann es nötig sein dasselbe Prüfprogramm dynamisch auszuführen, um sicher zu sein daß alle Bausteine im System bei Systemgeschwindigkeit arbeiten. An diesem Punkt kann es nötig sein einen Schreibvorgang (STA, STX, STY usw.), in die Schleife einzubauen, um ein Synchronisationssignal zu haben.

Bei der Untersuchung der dynamischen Arbeitsweise von ROM sollten zunächst jede Adress- und Datenbus-Leitung direkt am Prozessor angesehen werden. Hier wird die Adresse erzeugt und die Daten müssen einer bestimmten Geschwindigkeit entsprechen. Wenn die Daten nicht zur rechten Zeit gültig sind, wird im nächsten Schritt untersucht, wo im Datenweg vom Adress-Eingang, durch das ROM und zurück zum Prozessor Datenbus die zu große Verzögerung entstanden ist. Zur richtigen Arbeitsweise des Prozessors muß dieser ganze Weg die Arbeitsgeschwindigkeit erreichen. Wenn die Verzögerungen zu groß sind, kann es nötig sein den Systemtakt zu verlangsamen, damit die Programmdaten den Prozessor rechtzeitig erreichen. Eine andere Lösung für dieses Problem ist die Verwendung des RDY - Signals um den Prozessor jedesmal für einen Zyklus anzuhalten, wenn er Daten oder Programm vom ROM holt. Obwohl an diesem Punkt die angesprochenen Probleme auftreten können, ist es sehr viel wahrscheinlicher daß ein Verdrahtungsfehler bewirkt, daß eine einzelne Adress- oder Datenleitung übermäßig belastet wird. Dadurch arbeitet sie langsam oder überhaupt nicht.

Dieses Problem kann normalerweise leicht durch Betrachtung jedes einzelnen Bauteils im Datenweg erkannt und behoben werden.

B. RAM:

Die Arbeitsweise von RAM in einem Mikroprozessor System kann auf sehr ähnliche Weise wie beim ROM überprüft werden. Die Ausführung eines Prüfschleifen -Programms für jedes RAM im System sollte ausreichend sein um die richtige Arbeitsweise zu bestätigen. Für jedes RAM sollte in der Prüfschleife ein Lese- und ein Schreibvorgang vorgesehen sein. Dadurch wird die Überprüfung von Datenübertragung in beide Richtungen möglich.

Während der Ausführung der Prüfschleife im Einzelzyklus hält der Prozessor nur bei den Schreibvorgängen in das RAM an. Das erlaubt jedoch eine statische Überprüfung der Logik zur Baustein-Auswahl und der Adress- und Datenleitungen. Der dynamische Lauf des Programms ermöglicht die Überprüfung der Daten und Adress-Signale an den RAM während des Schreibvorganges. Es ist zu prüfen ob sie innerhalb der Daten für die im System verwendeten RAM liegen und das die Gesamtdurchlaufzeit durch den Weg Adresse, RAM, und Datenbus innerhalb der Daten für den Prozessor während des Schreibvorgangs liegen. Das wahrscheinlichste Problem ist, wie mit ROM, ein Verdrahtungsfehler, der einen bestimmten Baustein falsch arbeiten läßt. Eine sorgfältige Überprüfung jedes Stiftes ermöglicht die Erkennung dieses Problems.

C. PIA:

Die Bausteine für periphere Schnittstellen (6520, 6530 usw.) können alle in der oben beschriebenen Weise geprüft werden. Da diese Bausteine jedoch auf viele verschiedene Arten arbeiten können, muß das Prüfprogramm sehr viel komplizierter als das für die ROM und die RAM sein. Gewöhnlich kann man die Prüfung nur auf die Funktionen beschränken, die tatsächlich im System benutzt werden.

Ein großer Teil der Arbeitsweise der Schnittstellen-Bausteine kann durch einen Schreibvorgang, gefolgt durch einen Lesevorgang für jedes Register auf dem Baustein überprüft werden. So ist eine vollständige Überprüfung der Datenwege zwischen Prozessor und den Bausteinen und die Überprüfung aller Baustein-Auswahlfunktionen möglich. Es kann eine genauere Untersuchung nötig sein, um festzustellen daß die Daten richtig an den Ausgangsstiften des Peripherie-Bausteins erscheinen und daß die Daten an den Eingängen richtig zum Prozessor übertragen werden. Dazu muß das periphere Subsystem, das der Prozessor treibt, abgeklemmt und von Hand Daten in die Eingänge gegeben werden. Eine getrennte Prüfung stellt die Gültigkeit der Ausgangsdaten fest.

Nachdem die System Hardware in den Einzelheiten untersucht worden ist, wie oben beschrieben, wird der Entwickler die Auffassung haben, daß sein System ordnungsgemäß arbeitet, sobald das Systemprogramm fehlerfrei ist. Bestätigung des Systemprogramms sollte durch eine abschnittsweise Überprüfung wie oben besprochen vor sich gehen. Jede Subroutine, Interruptroutine usw. sollte getrennt untersucht werden. Sie können dann zusammengesetzt werden, um die größeren Betriebsroutinen für die Peripherie, arithmetische Routinen usw. darzustellen, die das System ausmachen. Das Endergebnis sollte ein arbeitsfähiges Programm sein, das in allen seinen Einzelheiten untersucht worden ist und das auf einem System läuft das voll funktionsfähig ist.

Anhang A

Zusammenfassung: Ausführung eines Einzelzyklus

Dieser Abschnitt enthält eine Übersicht der Daten auf Adress- und Datenbus für jeden Zyklus der verschiedenen Prozessor-Befehle. Die Zustände beim Durchlaufen des Programms in einzelnen Zyklen sind angegeben.

Der Prozessor hält nicht in einem Zyklus an, in dem R/W = 0 ist (Schreibzyklus). Er geht statt dessen bis zum nächsten Lesezyklus und hält dort an. Aus diesem Grund können einige Befehle kürzer erscheinen als hier angegeben.

Alle Befehle fangen mit T0 an und dem Holen des OP CODE setzen sich die nötige Zahl von Zyklen bis zum nächsten T0 und dem Holen des nächsten OP CODE fort.

Die Terminologie in diesem Anhang wird in der Programmier-Fibel erläutert. Zum leichten Nachsehen sind die Begriffe hier noch einmal aufgeführt.

OP CODE — Das erste Byte des Befehls. Es enthält den Operator und die Adress-Art.

OPERAND — Die Daten, mit denen der Befehl im OP CODE ausgeführt wird.

BASE ADRESS — Die Adresse, die in den indizierten Adress-Arten die Stelle im Speicher angibt, auf den sich die Indizierung bezieht. Das höherwertige Byte der Base Adress (AB 08-AB15) ist BAH (Base Adress High) und das niederwertige ist BAL (Base Adress Low).

EFFECTIVE ADRESS — Der Ort, an dem die Daten im Speicher liegen. Diese Adresse kann im Fall der Adressierarten PAGE Null und Absolut direkt geladen oder, wie bei den Index-Vorgängen berechnet werden. Das höherwertige Byte der effektiven Adresse ist ADH (AB 08-AB 15) und das niederwertige ADL (AB 00-AB 07)

INDIRECT ADRESS — Die Adresse im Operanden von Befehlen mit (Indirect), Y, die das niederwertige Byte der Base Adress enthält. IAH und IAL sind die entsprechenden Bytes.

JUMP ADRESS — Der als Ergebnis eines Sprung-Befehls (Jump) in den Programmzähler geladene Wert.

A. 1. SINGLE BYTE INSTRUCTIONS

ASL	DEX	NOP	TAX	TYA
CLC	DEY	ROL	TAY	
CLD	INX	SEC	TSX	
CLI	INY	SED	TXA	
CLV	LSR	SEI	TXS	

Die Ausführung dieser Ein-Byte-Befehle erfordert zwei Zyklen. Während des zweiten Zyklus wird die Adresse des nächsten Befehls in der Programm folge auf den Adressbus gelegt. Der OP CODE, der im zweiten Zyklus auf dem Datenbus liegt, wird nicht beachtet. Dieser Befehl wird im folgenden Zyklus geholt und dann dekodiert und ausgeführt.
Die Befehle ASL, ROL und LSR beziehen sich auf die Adress-Art Akkumulator.

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T0	PC + 1	OP CODE	1	Next Instruction

A. 2. INTERNAL EXECUTION ON MEMORY DATA

ADC	CMP	EOR	LDY
AND	CPX	LDA	ORA
BIT	CPY	LDX	SBC

Die oben aufgeführten Befehle werden im Prozessor mit Daten von der effektiven Adresse ausgeführt. Der gesamte Vorgang hat drei Schritte. Im ersten Schritt (ein Zyklus) wird der OP CODE geholt. Der zweite (null bis vier Zyklen) ist die Berechnung der effektiven Adresse. Im letzten Schritt werden Daten von der effektiven Adresse geholt. Der Befehl wird ausgeführt, während der nächste Befehl geholt und dekodiert wird.

A. 2.1. Immediate Addressing (2 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.2. Zero Page Addressing (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Effective Address
T2	00, ADL	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.3. Absolute Addressing (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order Effective Address byte
T2	PC + 2	ADH	1	Fetch high order Effective Address byte
T3	ADH, ADL	Data	1	Fetch Data
T0	PC + 3	OP CODE	1	Next Instruction

A. 2.4. Indirect, X Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + X	ADL	1	Fetch low order byte of Effective Address
T4	00, BAL + X + 1	ADH	1	Fetch high order byte of Effective Address
T5	ADH, ADL	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.5. Absolute, X or Absolute, Y Addressing (4 or 5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + index register	Data*	1	Fetch data (no page crossing)
	ADH: BAH + C			Carry is Ø or 1 as required from previous add operation
T4*	ADL: BAL + index register	Data	1	Fetch data from next page
	ADH: BAH + 1			
T0	PC + 3	OP CODE	1	Next Instruction

*If the page boundary is crossed in the indexing operation, the data fetched in T3 is ignored. If page boundary is not crossed, the T4 cycle is bypassed.

A. 2.6. Zero Page, X or Zero Page, Y Addressing Modes (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + index register	Data	1	Fetch Data (no page crossing)
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.7. Indirect, Y Addressing Mode (5 or 6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch low order byte of Base Address
T3	00, IAL + 1	BAH	1	Fetch high order byte of Base Address
T4	ADL: BAL + Y	Data*	1	Fetch Data from same page
	ADH: BAH + C			Carry is 0 or 1 as required from previous add operation
T5*	ADL: BAL + Y	Data	1	Fetch Data from next page
	ADH: BAH + 1			
T0	PC + 2	OP CODE	1	Next Instruction

*If page boundary is crossed in indexing operation, the data fetch in T4 is ignored. If page boundary is not crossed, the T5 cycle is bypassed.

A. 3. STORE OPERATIONS

STA
STX
STY

Die einzelnen Schritte bei den Speichervorgängen (Store Operations) sind denen in der vorhergehenden Gruppe sehr ähnlich (interne Ausführung mit Daten aus dem Speicher). Bei Speichervorgängen wird das Holen von Daten durch einen Schreibzyklus (WRITE, R/W = 0).
Bei Index-Vorgängen gibt es keine Überlappung und keine Verkürzung der Befehlszeit.

A. 3.1. Zero Page Addressing (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Zero Page Effective Address
T2	00, ADL	Data	0	Write internal register to memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.2. Absolute Addressing (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Effective Address
T2	PC + 2	ADH	1	Fetch high order byte of Effective Address
T3	ADH, ADL	Data	0	Write internal register to memory
T0	PC + 3	OP CODE	1	Next Instruction

A. 3.3. Indirect, X Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + X	ADL	1	Fetch low order byte of Effective Address
T4	00, BAL + X + 1	ADH	1	Fetch high order byte of Effective Address
T5	ADH, ADL	Data	0	Write internal register to memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.4. Absolute, X or Absolute, Y Addressing (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + index register ADH: BAH + C	Data (Discarded)	1	
T4	ADH, ADL	Data	0	Write internal register to memory
T0	PC + 3	OP CODE	1	Next Instruction

A. 3.5. Zero Page, X or Zero Page, Y Addressing Modes (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	ADL: BAL + index register	Data	0	Write internal register to memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.6. Indirect, Y Addressing Mode (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch low order byte of Base Address
T3	00, IAL + 1	BAH	1	Fetch high order byte of Base Address
T4	ADL: BAL + Y	Data (Discarded)	1	
	ADH: BAH			
T5	ADH, ADL	Data	0	Write Internal Register to memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 4. READ--MODIFY--WRITE OPERATIONS

ASL	LSR
DEC	ROL
INC	ROR

The Read--Modify--Write operations involve the loading of operands from the operand address, modification of the operand and the resulting modified data being stored in the original location.

Note: The ROR instruction will be available on MCS650X microprocessors after June, 1976.

A. 4.1. Zero Page Addressing (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Page Zero Effective Address
T2	00, ADL	Data	1	Fetch Data
T3	00, ADL	Data	0	
T4	00, ADL	Modified Data	0	Write modified Data back to memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 4.2. Absolute Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Effective Address
T2	PC + 2	ADH	1	Fetch high order byte of Effective Address
T3	ADH, ADL	Data	1	
T4	ADH, ADL	Data	0	
T5	ADH, ADL	Modified Data	0	Write modified Data back into memory
TØ	PC + 3	OP CODE	1	Next Instruction

A. 4.3. Zero Page, X Addressing (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded	1	
T3	ADL: BAL + X (without carry)	Data	1	Fetch Data
T4	ADL: BAL + X (without carry)	Data	0	
T5	ADL: BAL + X (without carry)	Modified Data	0	Write modified Data back into memory
TØ	PC + 2	OP CODE	1	Next Instruction

A. 4.4. Absolute, X Addressing (7 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch low order byte of Base Address
T2	PC + 2	BAH	1	Fetch high order byte of Base Address
T3	ADL: BAL + X	Data (Discarded)	1	
	ADH: BAH + C			
T4	ADL: BAL + X	Data	1	Fetch Data
	ADH: BAH + C			
T5	ADH, ADL	Data	0	
T6	ADH, ADL	Modified Data	0	Write modified Data back into memory
T0	PC + 3	OP CODE	1	New Instruction

A. 5. MISCELLANEOUS OPERATIONS

BCC	BRK	PHP
BCS	BVC	PLA
BEQ	BVS	PLP
BMI	JMP	RTI
BNE	JSR	RTS
BPL	PHA	

A. 5.1. Push Operation--PHP, PHA (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Pointer*	Data	0	Write Internal Register into Stack
T0	PC + 1	OP CODE	1	Next Instruction

*Subsequently referred to as "Stack Ptr."

A. 5.2. Pull Operations--PLP, PLA (4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	Data	1	Fetch Data from Stack
T0	PC + 1	OP CODE	1	Next Instruction

A. 5.3. Jump to Subroutine--JSR (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Subroutine Address
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr.	PCH	0	Push high order byte of program counter to Stack
T4	Stack Ptr. - 1	PCL	0	Push low order byte of program counter to Stack
T5	PC + 2	ADH	1	Fetch high order byte of Subroutine Address
T0	Subroutine Address (ADH, ADL)	OP CODE	1	Next Instruction

A. 5.4. Break Operation--(Hardware Interrupt)-BRK (7 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch BRK OP CODE (or force BRK)
T1	PC + 1 (PC on hardware interrupt)	Data (Discarded)	1	
T2	Stack Ptr.	PCH	0	Push high order byte of program counter to Stack
T3	Stack Ptr. - 1	PCL	0	Push low order byte of program counter to Stack
T4	Stack Ptr. - 2	P	0	Push Status Register to Stack
T5	FFFE (NMI-FFFA) (RES-FFFC)	ADL	1	Fetch low order byte of interrupt vector
T6	FFFF (NMI-FFFB) (RES-FFFD)	ADH	1	Fetch high order byte of interrupt vector
T0	Interrupt Vector (ADH, ADL)	OP CODE	1	Next Instruction

A. 5.5. Return from Interrupt-RTI (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	Data	1	Pull P from Stack
T4	Stack Ptr. + 2	Data	1	Pull PCL from Stack
T5	Stack Ptr. + 3	Data	1	Pull PCH from Stack
T0	PCH, PCL	OP CODE	1	Next Instruction

A. 5.6. Jump Operation--JMP

A.5.6.1. Absolute Addressing Mode (3 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch low order byte of Jump Address
T2	PC + 2	ADH	1	Fetch high order byte of Jump Address
T0	ADH, ADL	OP CODE	1	Next Instruction

A.5.6.2. Indirect Addressing Mode (5 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch low order byte of Indirect Address
T2	PC + 2	IAH	1	Fetch high order byte of Indirect Address
T3	IAH, IAL	ADL	1	Fetch low order byte of Jump Address
T4	IAH, IAL + 1	ADH	1	Fetch high order byte of Jump Address
T0	ADH, ADL	OP CODE	1	Next Instruction

A. 5.7. Return from Subroutine--RTS (6 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	PCL	1	Pull PCL from Stack
T4	Stack Ptr. + 2	PCH	1	Pull PCH from Stack
T5	PCH, PCL (from Stack)	Data (Discarded)	1	
T0	PCH, PCL + 1	OP CODE	1	Next Instruction

A. 5.8. Branch Operation--BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS (2, 3, or 4 cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TO	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Offset	1	Fetch Branch Offset
T2*	PC + 2 + offset (w/o carry)	OP CODE	1	Offset Added to Program Counter
T3**	PC + 2 + offset (with carry)	OP CODE	1	Carry Added

*Skip if branch not taken

**Skip if branch not taken; skip if branch operation doesn't cross page boundary.