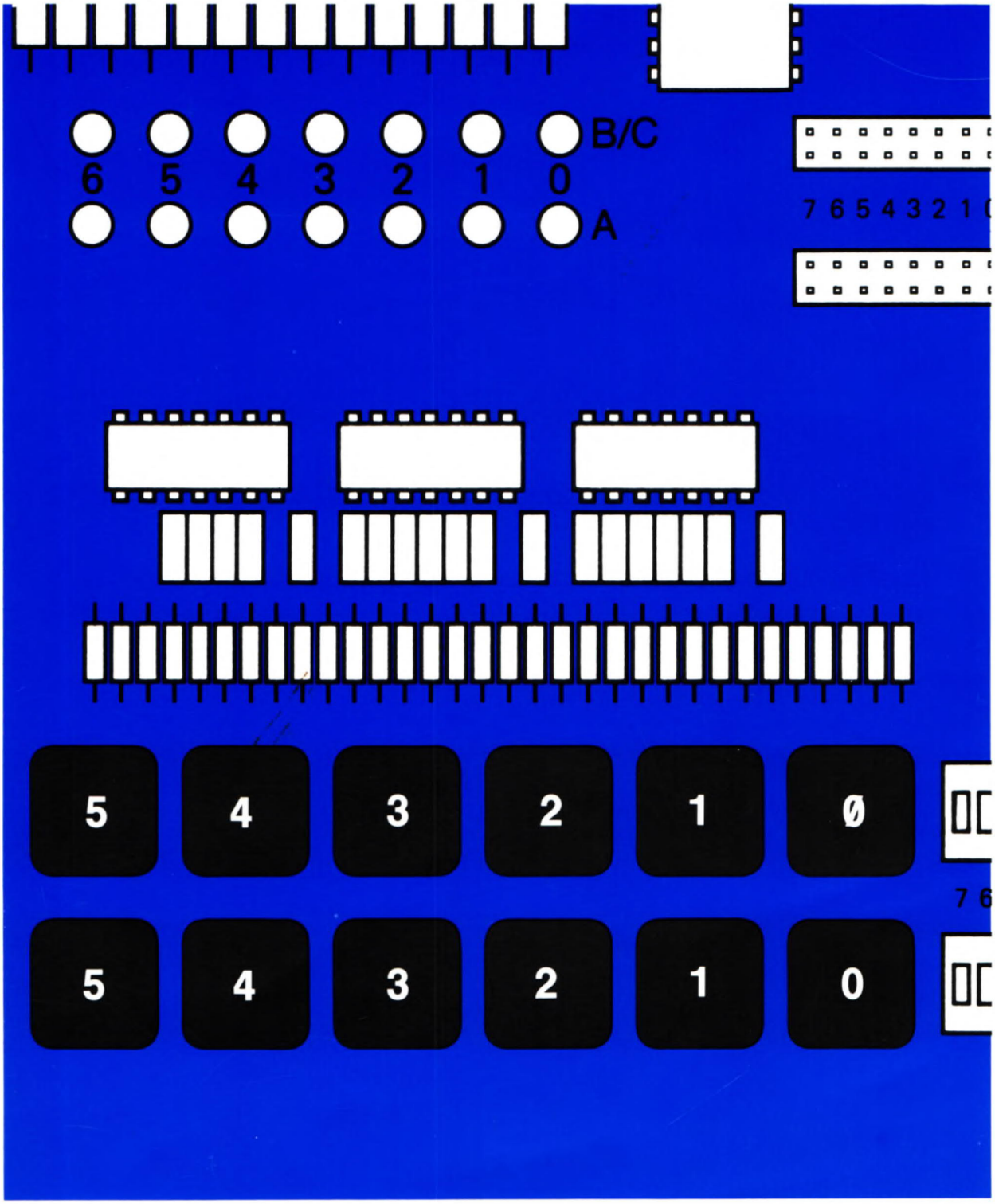
Peripherie-Bausteine

8255 · Z 80 PIO · CTC · Z 80 Interrupt



Christiani

PERIPHERIE-BAUSTEINE 1

Verfasser: Hans Fischer · Herausgeber: R. Christiani

Inhaltsverzeichnis

Seite

Vorwort 1

H **Hardware**

Peripherie und Interface 3
Was ist Peripherie 3
Was ist ein Interface 5

Die Peripherie-Leiterplatte 11
Z 80 PIO und Z 80 CTC auf dem Micro-professor 11
Anschluß der Peripherie-Leiterplatte an den Micro-Professor 12
Die Funktion der Interface-Leiterplatte 13

Die Adressierung der Z 80 PIO 23

S **Software**

Die Programmierung der Z 80 PIO 17
Die Betriebsarten der Z 80 PIO 18
Das Betriebsart-Steuerwort 22

Die Programmierung der Betriebsart 3 31
Das I/O-Register-Steuerwort 32
Einlesen und Ausgeben von Daten über einen PIO-Port 33

Versuche auf der Peripherie-Leiterplatte 34
Von Hand gesteuerte Daten-Ausgabe 34
Gesteuertes Lauflicht 38

Fehler: 1/24, 1/28, -/3, 1/44, 1/39,

Übungen

Seite

Lösungen der im Text gestellten Aufgaben 43

Prüfungsaufgaben

Hinweise 45
Aufgaben 47

Programm-Listen

Das Assembler-Format 49
Programm: Serielle Ausgabe 51
Programm: Flipflops . . . . . .
Programm: Lauflicht . . . . . .

1.A.840302

Vorwort

Das Thema dieses Lehrgangs ist die Beschreibung und Handhabung von Bausteinen, die speziell für die Vermittlung des Datenverkehrs eines Mikroprozessors mit seiner Umwelt (**Peripherie**) eingerichtet sind. Diese Bausteine werden als **Interface-Bausteine** bezeichnet.

Das Prinzip der Programmierung eines Mikroprozessors in Maschinensprache und die Darstellung von Befehlen in mnemonischen Codes sowie die Hardware-Struktur eines arbeitsfähigen Minimal-Systems sollen in diesem Lehrgang nicht behandelt werden. Diese Kenntnisse vermittelt der Lehrgang „Mikroprozessortechnik" am Beispiel des Mikroprozessors Z 80 im Micro-Professor-System. Dabei wird die CPU vorzugsweise so betrachtet, als handle es sich um einen Mikroprozessor des Typs 8085.

Grundkenntnisse der Hardware und der Programmierung des Mikroprozessors 8085 sind also hinreichende Voraussetzungen zum Durcharbeiten dieses Lehrgangs. Sie sollten jedoch mit den mnemonischen Codes für die Befehle des Z 80-Mikroprozessors vertraut sein.

In den Lehrbriefen finden Sie des öfteren Rückverweise auf Textstellen im Lehrgang Mikroprozessortechnik. Wenn Sie diesen Lehrgang durchgearbeitet haben, dann können Sie eventuelle Gedächtnislücken durch Nachlesen an der genannten Stelle ausfüllen.

Die Teilnahme am Lehrgang Mikroprozessortechnik ist aber keineswegs Voraussetzung dafür, daß Sie den hier vorliegenden Lehrgang mit Gewinn bearbeiten können.

In diesem Lehrgang wird der zur Z 80-Familie gehörende Interface-Baustein **Z 80 PIO** und der Timer **Z 80 CTC** sowie der zur 8085-Familie gehörende Interface-Baustein **8255** vorgestellt.

Außerdem wird das Interrupt-Prinzip des Mikroprozessors Z 80 im Zusammenhang mit dem Betrieb von Peripherie-Bausteinen behandelt.

Die im Lehrgang beschriebenen Versuche werden am Micro-Professor-System durchgeführt. Dieses System sollte also beim Durcharbeiten des Lehrgangs zur Verfügung stehen.

In der Materialsendung zu diesem Lehrbrief finden Sie die in der Grundausrüstung des Micro-Professors noch nicht bestückten Bausteine Z 80 PIO und den Timer CTC. Wir zeigen Ihnen im Lehrbrief, wie diese ICs in den Micro-Professor eingebaut werden.

Außerdem enthält die Materialsendung eine Zusatz-Leiterplatte mit Tasten und Leuchtdioden, sowie einem für Versuche zur Verfügung stehenden Interface-Baustein 8255. Über die Tasten auf der Leiterplatte können Signale in das Micro-Professor-System eingegeben werden. Die Leuchtdioden zeigen Signale an, die das System – vom Programm gesteuert – nach außen abgibt.

Über eine Klemmleiste können Ihrem System von irgendwelchen fremden Geräten passende Signale geliefert werden. Auf dem gleichen Wege können Sie solche Geräte auch vom Micro-Professor steuern lassen. Zwei kleine Relais auf der Zusatz-Leiterplatte erlauben es, fremde Geräte galvanisch getrennt anzuschließen.

Wie das alles gemacht wird, und vor allem, wie der Micro-Professor für eine solche Daten-Kommunikation programmiert wird, das wollen wir Ihnen in diesem Lehrgang zeigen.

Die Programme zu den Versuchen im Lehrgang werden in der Form angeschrieben, wie sie bei der Verwendung eines Assemblers benutzt wird. In die Besonderheiten dieser Form führen wir Sie vor dem Lehrbrief-Abschnitt mit den Programmen ein.

Die Abschnitte des Lehrgangs sind mit Griffmarken gekennzeichnet:

H	Hardware	S	Software
P	Prüfungsaufgaben	Ü	Übungen
L	Programm-Listen	T	Tabellen

Die Listen für die Versuchs-Programme fassen wir in einem eigenen Abschnitt zusammen, damit der Lehrstoff selbst übersichtlich bleibt. In dieser Zusammenfassung können Sie später die einzelnen Programme leicht wieder auffinden.

Die Eigenschaften der vorgestellten Peripherie-Bausteine werden jeweils dann in einer Tabelle zusammengefaßt, wenn Sie die Handhabung eines Bausteins kennengelernt haben.

Innerhalb eines Lehrbriefs wechseln Abschnitte mit verschiedenen Griffmarken je nach Fortschritt des Lehrstoffs ab. Nach dem Durcharbeiten des Lehrgangs können die mit gleichen Griffmarken bezeichneten Abschnitte zusammengefaßt abgelegt werden.

PERIPHERIE-BAUSTEINE 2

Verfasser: Hans Fischer · Herausgeber: R. Christiani

Inhaltsverzeichnis Seite

Die Themen des Lehrbriefs 1

H **Hardware**

Die Centronics-Schnittstelle 3
Das Handshake bei der Centronics-Schnittstelle . . . 4
Anschluß eines Druckers mit Centronics-Schnittstelle 7
Initialisierung der PIO für eine Centronics-Schnittstelle 8
Die Codierung von Druckzeilen 10
Die Centronics-Druck-Routine 12

Interrupt-Programmierung der Z 80 CPU 29
Die Interrupt-Modes des Z 80 29
Das I-Register 31
Interrupt-Prioritäten 32

S **Software**

Die Interrupt-Struktur des Z 80 15
Ein einfaches Programm mit Interrupt 15
Aufruf einer Interrupt-Service-Routine 18

Die Interrupt-Programmierung der Z 80 PIO 21

Ein Morse-Programm mit Interrupts 39
Das Prinzip des Morse-Programms 39
Interrupts im Morse-Programm 40

Ü **Übungen**

Lösungen der im Text gestellten Aufgaben 47

P **Prüfungsaufgaben**

Aufgaben 51

L **Programm-Listen**

Programm: Hex-Dump im Micro-Professor 53
Programm: Lauflicht mit Interrupt 59
Programm: Lauflicht mit Abfrage 63
Programm: Morsen mit Interrupt 67

TECHNISCHES LEHRINSTITUT DR.-ING. P. CHRISTIANI KONSTANZ

Fehler: 2/11, 2/14, 2/20, 2/24, 2/43, 2/51, 2/30

Die Themen des Lehrbriefs

Im Lehrbrief 1 unseres Lehrgangs über Peripheriebausteine haben wir Ihnen die Grundfunktionen der Z 80 PIO vorgestellt. Sie haben gesehen, daß es für die beiden Ports der PIO unterschiedliche Betriebsarten gibt, die unabhängig voneinander programmiert werden können.

Für die meisten Anwendungen ist die Betriebsart 3 am interessantesten, und mit dieser Betriebsart haben Sie auch die Versuche an der Peripherie-Leiterplatte durchgeführt.

Die Betriebsarten Ø und 1 sind einander sehr ähnlich; sie unterscheiden sich nur darin, daß die Betriebsart Ø für das Aussenden von Daten zuständig ist und die Betriebsart 1 für das Einlesen von Daten (Seiten S 3 und S 4). Die Betriebsart 2 ist Spezialfällen vorbehalten (Seite S 5).

Wir haben bereits auf der Seite S 7 darauf hingewiesen, daß z. B. bei der Betriebsart 1 das von der Peripherie ausgesandte STROBE-Signal dazu benutzt wird, ein von der CPU gerade in Arbeit befindliches Programm zu unterbrechen und eine Einlese-Routine für das Byte aufzurufen, das die Peripherie mit dem STROBE-Signal angemeldet hat.

Offenbar spielt also das Interrupt-System der Z 80-CPU beim Einsatz der Z 80 PIO eine wichtige Rolle. Sie werden noch sehen, daß das auch beim Z 80 CTC der Fall ist. Das Arbeiten mit Interrupts macht eine Art der Programmierung möglich, mit der manche Aufgaben auf verblüffend elegante Weise gelöst werden können.

Sie werden sich in diesem Lehrbrief eingehend mit dem Interrupt-System der Z 80-CPU beschäftigen, insbesondere natürlich im Zusammenhang mit den in diesem Lehrgang interessierenden Peripherie-Bausteinen. Welche besonderen Möglichkeiten sich dadurch ergeben, können Sie dann in ein paar recht hübschen Versuchen feststellen.

Ehe wir Ihnen die Funktionen des Z 80-Interrupt-Systems zeigen, stellen wir Ihnen am Anfang dieses Lehrbriefs eine Routine vor, die Sie vermutlich nicht ausprobieren können, die Ihnen aber bei der weiteren Beschäftigung mit Mikroprozessor-Systemen nützlich sein kann. Es ist eine Routine, die den Betrieb eines Druckers mit einer sogenannten Centronics-Schnittstelle an einem Mikroprozessor-System gestattet.

Wir beschreiben diese Routine zunächst für die Betriebsart 3 der Z 80 PIO. Später stellen wir Ihnen noch zwei weitere Versionen dieser Routine vor, in denen von den Möglichkeiten des Interrupt-Betriebes Gebrauch gemacht wird. Die dritte Version zeigt Ihnen, wie sehr elegant solch eine Routine für die Betriebsart Ø der PIO geschrieben werden kann.

Wenn Sie zufällig einen Drucker mit Centronics-Schnittstelle verfügbar haben, dann können Sie diesen Drucker über den PIO CTC I/O BUS an Ihren Micro-Professor anschließen und drucken lassen.

Sie werden in diesem Lehrbrief aber keineswegs nur Programme vorfinden, die Sie nicht selbst ausprobieren können. Das Arbeiten mit

Interrupts zeigen wir Ihnen an einigen recht instruktiven Versuchen, und Sie werden erkennen, daß die Z 80 PIO erst dann ihr volles Können zeigt, wenn die Möglichkeiten des Interrupts genutzt werden.

Die Z 80 PIO ist ein Mitglied der Z 80-Mikroprozessor-Familie. Alle zu dieser Familie gehörenden Bausteine (auch der im folgenden Lehrbrief behandelte CTC-Baustein!), sind so aufeinander abgestimmt, daß das Interrupt-System der CPU voll genutzt werden kann. Was es damit auf sich hat, soll ebenfalls das Thema dieses Lehrbriefs sein.

Die Beschäftigung mit dem Interrupt-System wird sich bei der späteren Beschreibung des CTC-Bausteins auszahlen, denn dieser Baustein ist nur dann wirklich sinnvoll einzusetzen, wenn Interrupt-Routinen verwendet werden.

PERIPHERIE-BAUSTEINE 3

Verfasser: Hans Fischer · Herausgeber: R. Christiani

Inhaltsverzeichnis

Seite

Die Themen des Lehrbriefs 1

H **Hardware**

Centronics-Schnittstelle mit Interrupt 3
Interrupt in der Betriebsart 3 3
Interrupt in der Betriebsart Ø 8

Aufbau und Anschluß des Z 80 CTC 19
Der interne Aufbau eines CTC-Kanals 19
Die Adressierung des Z 80 CTC 22
Interrupts über den CTC 25

S **Software**

Der Z 80 CTC 11
Was macht man mit dem CTC? 11
Die Programmierung des CTC 14

Die Programmierung des Z 80 CTC 27
Das Channel Control Word 27
Das Time Constant Word 31
Das Interrupt Control Word 33
Ein einfaches Programm mit CTC-Interrupt 36

PIO- und CTC-Interrupts 37
Eine schaltbare Sekunden-Uhr 37
Interrupt-Lauflicht und Sekunden-Uhr 40
Serielle Daten-Ausgabe mit CTC-Interrupt 41

Ü **Übungen**

Lösungen der im Text gestellten Aufgaben 45

P **Prüfungsaufgaben**

Aufgaben 49

L **Programm-Listen**

Programm: Hex-Dump mit Interrupt 51
Programm: Ereignis-Zähler 57
Programm: Interrupt-Lauflicht 59
Programm: Lauflicht mit Sekunden 63
Programm: Serielle Ausgabe mit Anzeige 71

TECHNISCHES LEHRINSTITUT DR.-ING. P. CHRISTIANI KONSTANZ

1.A.840502

Die Themen des Lehrbriefs

Der Titel unseres Lehrgangs verspricht die Beschäftigung mit Peripherie-Bausteinen – und sicher haben Sie nicht den Eindruck, daß dieses Thema bisher zu kurz gekommen ist. Mit der Programmierung und Handhabung der Z 80 PIO sollten Sie inzwischen recht gut vertraut sein.

In diesem Lehrbrief stellen wir Ihnen vorzugsweise den Z 80 CTC vor. Es handelt sich um einen Baustein, dessen Eigenschaften ein klein wenig anders geartet sind als die solcher Peripherie-Bausteine, die rein auf die Vermittlung des Datenverkehrs zwischen der CPU und ihrer Peripherie ausgerichtet sind. Aber Sie werden ja sehen – wir wollen nicht vorgreifen.

Insbesondere der zweite Lehrbrief dieses Lehrgangs hat Sie aber bestimmt auch davon überzeugt, daß es nur eine halbe Sache wäre, Peripherie-Bausteine einzusetzen, ohne von deren komfortablen Möglichkeiten der Interrupt-Programmierung Gebrauch zu machen. Gerade beim Z 80-System wurde Wert darauf gelegt, Interrupts in recht einfacher Weise einsetzen zu können.

Der Unterschied zwischen normaler Programmierung und der Verwendung von Interrupts läßt sich sehr schön an einer Centronics-Schnittstelle zeigen. Allein mit den experimentellen Mitteln dieses Lehrgangs können Sie zwar die Centronics-Schnittstelle nicht ausprobieren. Wenn Sie sich aber eingehender mit Mikroprozessoren beschäftigen, dann taucht das Problem eines Drucker-Anschlusses mit großer Sicherheit einmal auf. Und es ist gut, wenn Sie dann Bescheid wissen.

Wir werden uns aus diesem Grunde gleich am Anfang dieses Lehrbriefs noch einmal mit der Centronics-Schnittstelle beschäftigen. Dabei zeigen wir Ihnen in zwei Schritten, daß die Interrupt-Programmierung zu wirklich eleganten Problem-Lösungen führt. Am Ende dieses Lehrbriefs kommen wir dann noch einmal auf eine Drucker-Schnittstelle zurück.

Sie werden sehen, daß auch mit einem vorwiegend für die Bit-parallele Daten-Kommunikation ausgelegten Peripherie-Baustein eine serielle Schnittstelle programmiert werden kann. Selbstverständlich wird auch dabei von den Möglichkeiten der Interrupt-Programmierung Gebrauch gemacht.

Hier sind ein paar grundsätzliche Gedanken über die Interrupt-Programmierung angebracht. – Die Beschäftigung mit der Z 80 PIO hat gezeigt, daß es keines großen Aufwandes bedarf, Interrupts zu verwenden. Keines großen Aufwandes – gewiß nicht. Aber ohne ein paar wenige zusätzliche ICs geht es nun doch nicht.

Letztlich ist es eine Frage der Aufgabenstellung und der Wirtschaftlichkeit, ob man Interrupts verwendet oder nicht. Manche Aufgaben lassen sich überhaupt nur dann lösen, wenn man Interrupts einsetzt. Sie sind dann eine Notwendigkeit, wenn zeitkritische Probleme vorliegen, wenn man sich also im Programm aus zeitlichen Gründen einfach keine Abfrage- oder Warte-Schleifen leisten kann. In solchen Fällen nimmt man gern den zusätzlichen Aufwand eines oder mehre-

rer ICs in Kauf. Auch auf den Einwand hin, daß sich dieser preisliche Aufwand mit der Stückzahl multipliziert.

Wenn sich eine Software-Aufgabe ohne die noch so eleganten Möglichkeiten der Interrupt-Programmierung lösen läßt, dann hat der Kaufmann das Sagen: Die Entwicklung der Software kostet nur einmal Geld. Solche Überlegungen haben sicher auch bei der Entwicklung des Micro-Professors eine Rolle gespielt. Die Abfrage seiner Tastatur und die Bedienung der Sieben-Segment-Anzeige hätte sich technisch sicher viel eleganter und interessanter mit einer Interrupt-Programmierung erledigen lassen. Hier hat man der konventionellen Lösung mit Abfrage- und Bedienungs-Schleifen den Vorzug gegeben. Die Praxis zeigt, daß auch ohne den Interrupt-Aufwand eine wirklich gute Lösung herausgekommen ist – und eine preiswerte dazu.

Zurück zu unserem Lehrbrief: Vorwiegendes Thema wird der Z 80 CTC sein. Bei der Vorstellung dieses Bausteins werden Sie sehen, daß er mehr noch als die PIO mit der Verwendung von Interrupts lebt. Wenn man sich also zum Einsatz dieses Bausteins entschließt, dann ergeben sich Möglichkeiten, die von der CPU allein nur durch großen zeitlichen (und oft auch programm-technischen) Aufwand erledigt werden könnten.

PERIPHERIE-BAUSTEINE 4

Verfasser: Edgar Hoch · Herausgeber: R. Christiani

Inhaltsverzeichnis

Seite

Hardware

Der Ein- und Ausgabebaustein 8255 1
Der Modus 0 1
Der Modus 1 1
Der Modus 2 1
Der Modus 3 2
Die interne Struktur des 8255 2
Die Sockelbeschaltung des 8255 4
Der Anschluß des 8255 auf der Peripherie-Leiterplatte 5
Der Adreßbus 5
Die Adreßcodierung 5
Die Ausgänge des 8255 6
Die Belegung der Anschlußklemmleiste 7
Die Betriebsart 0 7
Einschreiben des Steuerworts in den 8255 8
Der Aufbau des Steuerworts 8

Der 8255 auf der Micro-Professor-Platine 17
Anschluß des 8255 auf der Micro-Professor-Platine 18
Die Funktion des Ports B 19
Die Funktion des Ports C 20
Die Funktion des Ports A 20

Die Betriebsart 1 des 8255 31
Port A und Port B arbeiten als Eingabeports 31
Die Bedeutung der Steuersignale an Port C 31
Die Interrupt-Verarbeitung 33
Das Impulsdiagramm bei der Dateneingabe 34
Bildung verschiedener Steuerwörter in der Betriebsart 1 35
Port A und Port B arbeiten als Ausgabeports in der Betriebsart 1 35
Die Bedeutung der Steuersignale an Port C 36
Das Impulsdiagramm bei der Datenausgabe 37

Die Betriebsart 2 des 8255 51
Die Bedeutung der Steuersignale an Port C 51
Die internen INTE-Flipflops 52
Der Port B und die freien Bits des Ports C 52

S **Software** Seite

Programmieren in Betriebsart 0 11
Binärer Zähler über Port C 13
Lauflicht mit Ausgabe auf Port A 13
Initialisierungsprogramm INIT 14
Das variable Zeitprogramm ZEIT 2 15

Programmierung des 8255 auf der Micro-Professor-Platine 21
Ein kleiner Fehler mit großer Wirkung 22
Interrupt mit Restart RST 38 23
Die Interrupt-Service-Routine 25
Vollständige Bedienung der Anzeigeeinheit 26
Der Anzeigepuffer 26
Der erste Programmteil 27
Die Unterprogramme 27
Das Umcodierungsunterprogramm HEX 7 29

Setzen einzelner Bits des Ports C in Betriebsart 0 39
Der Aufbau des Steuerworts für Bit-Manipulationen am Port C 39
Bedeutung des Steuerworts in der Betriebsart 1 . . . 41
Sperren oder Freigeben der INTE-Flipflops in der Betriebsart 1 41
Programmieren in der Betriebsart 1 — Dateneingabe 44
Programmieren in der Betriebsart 1 — Datenausgabe 48
Die Interrupt-Service-Routine 49

Ü **Übungen**

Lösungen der im Text gestellten Aufgaben 53

P **Prüfungsaufgaben**

Aufgaben 55

L **Programm-Listen**

Binärer Zähler über Port C 57
Lauflicht 58
Interrupt-Zähler mit Anzeige 59
Anzeigeprogramm mit dem 8255 auf der Micro-Professor-Platine 60
Bit-Manipulationen über Port C des 8255 62
Testprogramm für 8255 in Betriebsart 1 63
Einfaches Programm für 8255 in Betriebsart 1 65
Ausgabe eines Speicherbereichs über den Port A des 8255 in der Betriebsart 1 67

1.A.840802

Lehrgang PERIPHERIE-BAUSTEINE

Verfasser: Hans Fischer, Edgar Hoch · Herausgeber: R. Christiani

Hardware H

Software S

Prüfungsaufgaben P

Übungen Ü

Tabellen T

Programm-Listen L

Lehrgang PERIPHERIE-BAUSTEINE Verfasser: Hans Fischer, Edgar Hoch
Herausgeber: R. Christiani

Hardware

Peripherie und Interface

Die Tatsache, daß Sie sich zum Durcharbeiten dieses Lehrgangs entschlossen haben, zeigt, daß Sie sicher bereits wissen, was es mit Interface-Bausteinen auf sich hat. Es ist aber eine gute Methode (die leider häufig vergessen wird), sich zunächst einmal zu einigen, was man unter den Dingen eigentlich versteht, die das Thema einer Unterhaltung sein sollen. Das soll im folgenden Abschnitt geschehen.

Was ist Peripherie?

Im allgemeinen Sprachgebrauch ist Peripherie alles das, was um ein zentrales Etwas herum angeordnet ist und irgendwie zu diesem Etwas gehört. Peripherie sind zum Beispiel die Vororte einer Stadt, oder es ist auch die Umgebung eines Menschen, also das, was man als Umwelt bezeichnen kann.

In der Mikroprozessortechnik ist das zentrale Etwas der Mikroprozessor selbst, also die CPU (**C***entral* **P***rocessing* **U***nit*). So gesehen liegt es nahe, jedes Bauteil in einem Mikroprozessor-System, das zusätzlich zur CPU eingebaut ist, als Peripherie zu bezeichnen. Eine solche Definition wäre aber einfach unpraktisch.

Bei der Definition des Begriffs Peripherie faßt man das zentrale Etwas ein wenig weiter: Zum „Etwas" gehören alle die Teile, die ein sogenanntes **Minimal-System** arbeitsfähig machen.

Eine CPU allein ist noch kein arbeitsfähiges Gebilde. Die CPU kann erst dann eine ihr gestellte Aufgabe lösen, wenn man ihr mitteilt, welche Arbeiten sie zur Lösung der Aufgabe ausführen muß. Diese Mitteilung besteht aus einem Programm, in dem die einzelnen Arbeitsschritte in ihrer zeitlichen Reihenfolge entsprechend den Fähigkeiten der CPU angegeben werden.

Das Programm zur Lösung einer Aufgabe wird in einem Speicher abgelegt, der somit unumgänglicher Bestandteil eines arbeitsfähigen Minimal-Systems ist. (Daß es sogenannte Ein-Chip-Mikrocomputer gibt, die aus einer CPU mit integriertem Speicher bestehen, braucht in diesem Zusammenhang nicht zu interessieren.)

Meist gehören zu einem Minimal-System noch einige wenige Gatter-Funktionen und vielleicht noch ein paar einfache Flipflops, die aber mit der grundsätzlichen Funktion des Minimal-Systems nichts zu tun haben.

Hier werde festgestellt, daß ein arbeitsfähiges Mikroprozessor-Minimal-System aus einer CPU und einem Speicher besteht. Der Speicher wiederum besteht aus einem Festspeicher, in dem das Programm abgelegt ist, und aus einem mehr oder weniger umfangreichen Schreib-Lese-Speicher, in dem die CPU Daten, die bei der Lösung einer Aufgabe anfallen, zwischenzeitlich ablegen kann.

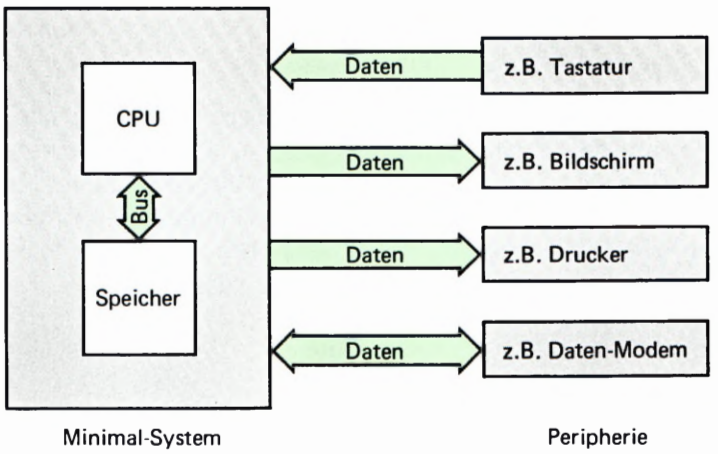


Bild H2.1
So sieht das Prinzip eines Mikroprozessor-Minimal-Systems mit angeschlossener Peripherie aus.

Das Minimal-System ist also das zentrale „Etwas"; alles, was außerhalb dieses Minimal-Systems angeordnet ist und in irgendeiner direkten Beziehung zu diesem Minimal-System steht, ist Peripherie.

Jetzt wird deutlich, daß ein Minimal-System ohne Peripherie ein ziemlich sinnloses Gebilde ist. Mit Hilfe eines Minimal-Systems soll ja immer eine bestimmte Aufgabe gelöst werden. Diese Aufgabe kann sich nicht auf das Minimal-System selbst beziehen, denn es wäre wirklich sinnlos, wenn sich das Minimal-System (wie es bei Verwaltungen manchmal den Anschein hat) selbst verwalten oder bearbeiten sollte.

Das Minimal-System löst eine Aufgabe durch das Verarbeiten von Daten. Im einfachsten Fall sind diese Daten einzelne elektrische LOW- oder HIGH-Signale. Meist sind die Daten aber Kombinationen aus jeweils acht elektrischen Einzelsignalen (Bytes), die dem Minimal-System gleichzeitig (parallel) über den Daten-Bus zugeführt werden, oder die das Minimal-System über den Daten-Bus abgibt.

Für das Minimal-System ist jedes „Ding" seiner Umgebung, das ihm im Rahmen einer gestellten Aufgabe Daten zur Verarbeitung liefert, und an das es die verarbeiteten Daten wieder abliefert, Peripherie (Bild H2.1).

Man sieht, daß Peripherie ein sehr umfangreicher Begriff ist. Peripherie kann zum Beispiel eine Tastatur sein, die Bytes an das Minimal-System liefert, wenn eine Taste betätigt wird. Daß diese Tastatur ihrerseits von einem Menschen bedient wird, ist für das Minimal-System ganz belanglos. Es „sieht" ja nur die Tastatur und sonst nichts.

Auch die Sieben-Segment-Anzeige des Mikro-Professors ist für das Minimal-System Peripherie. Im Gegensatz zur Tastatur, die ein Daten-Sender ist, ist die Sieben-Segment-Anzeige ein Daten-Empfänger. Auch hier interessiert sich das Minimal-System für den Menschen, der diese Anzeige letztlich abliest, überhaupt nicht.

Ein an das Cassetten-Interface angeschlossener Audio-Cassetten-Recorder kann sowohl Daten-Empfänger als auch Daten-Sender sein, denn er kann vom Minimal-System gelieferte Daten auf einer Cassette abspeichern, aber auch vorher abgespeicherte Daten wieder an das System senden.

Was ist ein Interface?

Die Daten, die ein Minimal-System an die Peripherie liefert, können von der CPU per Programm immer so aufbereitet werden, daß sie für die jeweilige Peripherie gerade richtig sind. Was damit gemeint ist, soll an einem Beispiel erläutert werden.

Bei der Verarbeitung von Texten in einem Mikroprozessor-System werden die Buchstaben des Alphabets, die Ziffern und die Satzzeichen jeweils durch ein Byte dargestellt. Dabei wird die Codierung, also die Zuordnung der Buchstaben, Ziffern und Satzzeichen zu bestimmten Bytes, entsprechend dem **A***merican* **S***tandard* **C***ode for* **I***nformation* **I***nterchange* (ASCII: Amerikanischer Standard Code für Informations-Austausch) vorgenommen. (Wir werden Ihnen diesen Code später noch vorstellen.)

Bei dieser Art der Text-Verarbeitung entspricht also jedem druckbaren Zeichen ein bestimmtes, aus acht Bit bestehendes Byte.

Unter Text-Verarbeitung wird im allgemeinen die Eingabe von Text in das Mikroprozessor-System über eine Art Schreibmaschinen-Tastatur verstanden, die Formatierung dieses Textes auf einem Bildschirm und die spätere Ausgabe des Textes auf einen Drucker.

In einem solchen System gehören die Tastatur, der Bildschirm und der Drucker zur Peripherie. Die Tastatur sendet die eingegebenen ASCII-Zeichen an das System, und das System sendet die verarbeiteten ASCII-Zeichen an den Bildschirm und an den Drucker.

Bei der Übertragung eines jeden ASCII-Zeichens zwischen Peripherie und System bzw. zwischen System und Peripherie müssen jeweils acht Bit übermittelt werden.

Für diese Übertragung gibt es zwei unterschiedliche Möglichkeiten:

1. Man kann jedem Bit eine eigene Leitung zuordnen. Die Übertragung geschieht also sozusagen auf dem verlängerten Datenbus des Systems. Für die Übertragung sind mindestens acht Datenleitungen und eine zusätzliche Leitung für das Bezugspotential notwendig. So kann ein Byte mit seinen acht Bit eins nach dem anderen zwischen der Peripherie und dem System transportiert werden. Man spricht dann von einer Bit-parallelen, Byte-seriellen Daten-Übertragung. Diese Art der Zeichen-Übertragung ist zwar naheliegend; wegen der vielen benötigten Leitungen ist sie aber recht aufwendig. (Bild H3.1.)

2. Man beschränkt sich für die Übertragung auf eine einzige Datenleitung. (Tatsächlich braucht man außer dieser Datenleitung und der Leitung für das Bezugspotential mindestens noch eine weitere Leitung, die wir aber der Einfachheit halber zunächst übersehen wollen. Wir kommen darauf noch zurück.)

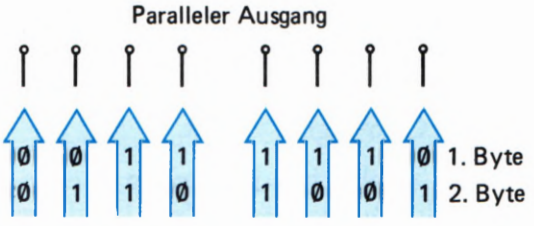


Bild H3.1
Bei der Bit-parallelen, Byte-seriellen Datenübertragung werden acht Datenleitungen benötigt.

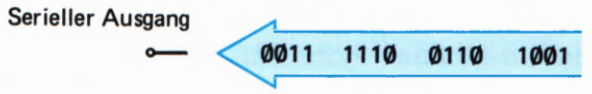


Bild H4.1
Die Bit-serielle, Byte-serielle Datenübertragung kann auf einer einzigen Datenleitung erfolgen.

Bei einer solchen Anordnung kann natürlich immer nur jeweils ein einziges der acht Bits eines Bytes für ein druckbares Zeichen übermittelt werden. Die acht Bits müssen also in zeitlicher Folge nacheinander auf die Reise geschickt werden. (Bild H4.1.)

Wenn diese Anordnung funktionieren soll, dann muß jedes Bit eines Bytes einzeln bereitgestellt und im richtigen Augenblick auf die Datenleitung geschaltet werden. Diese Art der Datenübertragung bezeichnet man als Bit-seriell, Byte-seriell.

Peripherie-Einheiten (Tastatur, Bildschirm, Drucker usw.) können sowohl für die erste oder auch für die zweite der hier beschriebenen Datenübertragungen eingerichtet sein.

Selbstverständlich kann das Mikroprozessor-System so programmiert werden, daß es seine Daten in der einen oder in der anderen Form von der Peripherie empfangen oder an die Peripherie aussenden kann. Mit der Verarbeitung der Daten (in diesem Fall also z. B. der Formatierung der druckbaren Zeichen) hat das aber wenig zu tun. Mit anderen Worten: Der CPU wird eine Aufgabe übertragen, für die sie eigentlich viel zu schade ist.

Ab der Seite L3 haben wir ein Programm aufgelistet, das die Bit-serielle, Byte-serielle Ausgabe von Daten nachbildet. – Um die interne Funktion dieses Programm brauchen Sie sich in diesem Zusammenhang nicht zu kümmern. (Wenn Sie sich dafür interessieren, dann geben Ihnen die eingetragenen Kommentare in der Programm-Auflistung entsprechende Hinweise. Wir haben allerdings im Programm einige Z80-spezifische Befehle verwendet, um das Programm nicht unnötig lang werden zu lassen.)

Der im Programm nachgebildeten Übertragung liegt – wie auch bei der in der Praxis verwendeten Übertragung – folgendes Prinzip zugrunde: Die Daten empfangende Peripherie teilt dem Daten sendenden Mikroprozessor-System über eine einadrige, getrennte Datenleitung durch ein CTS-Signal (*Clear To Send:* Du darfst senden) mit, daß sie bereit ist, ein Byte zu empfangen. Wenn das System dieses Signal empfängt, dann schaltet es auf die normalerweise im HIGH-Bereich liegende Sende-Datenleitung für eine definierte Zeit in LOW-Signal als sogenanntes Start-0-Bit. Nach Ablauf der definierten Zeit des Start-0-Bits beginnt die Aussendung der acht Daten-Bits des zu übertragenden Bytes in der Reihenfolge Bit Nr. 0 bis Bit Nr. 7. Einem 0-Bit ist ein LOW-Signal auf der Sende-Datenleitung zugeordnet, einem 1-Bit ein HIGH-Signal. Jedes dieser Signale bleibt jeweils gerade so lange stehen wie vorher das Start-0-Bit.

Nach der Aussendung der acht Daten-Bits eines Bytes werden ein oder zwei Stop-1-Bits ausgesendet. Erst dann kann die Übertragung des Start-0-Bits für das nächste Byte erfolgen, vorausgesetzt, daß die Peripherie das mit einem CTS-Signal zuläßt.

Im Bild H5.1 ist der Zyklus für das serielle Aussenden eines Bytes schematisch dargestellt. Die definierte Zeit, für die ein Signal jeweils

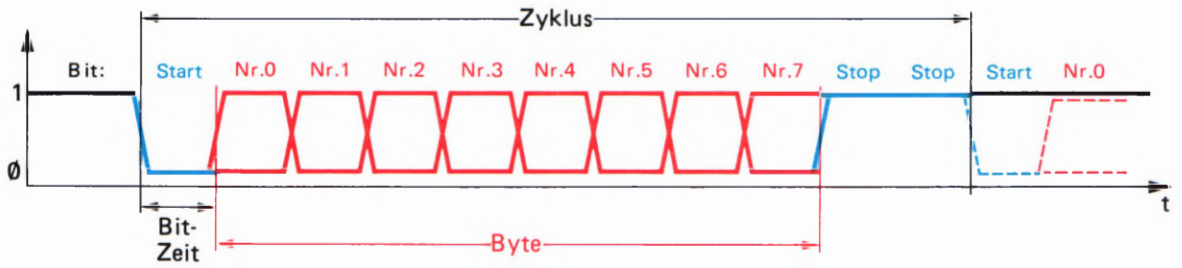


Bild H5.1
Der Zyklus zur Bit-seriellen Datenübertragung besteht aus einem Start-0-Bit, den 8 Daten-Bits und meist 2 Stop-1-Bits.

auf der Datenleitung steht, haben wir als Bit-Zeit bezeichnet. Die Geschwindigkeit, mit der ein Byte übertragen wird, hängt von der Bit-Zeit ab. In unserem Programm haben wir die Bit-Zeit recht lang gewählt, damit der Vorgang gut verfolgt werden kann. (In der Praxis liegt sie bei etwa 13 ms bei sehr langsam arbeitenden Systemen.)

Lesen Sie bitte zunächst die einführenden Anmerkungen auf der Seite L 1 zur Form unserer Programm-Auflistung, und geben Sie dann das Programm über die Tastatur des Micro-Professors ein.

Nach dem Start des Programms mit der Taste GO verlischt die Anzeige; es leuchtet nur der Dezimal-Punkt der rechten Anzeige-Stelle. Dieser Punkt imitiert den Signal-Pegel der Sende-Datenleitung (anfangs HIGH).

Das CTS-Signal der Peripherie, für welche die ausgesendeten Daten bestimmt sind, können Sie mit der Taste USER KEY ganz links in der unteren Reihe des Tastenfeldes imitieren. Sobald diese Taste betätigt wird, beginnt der Micro-Professor mit der Aussendung des ersten Bytes des mit ANFAD adressierten Speicher-Bereichs. Im Versuch ist es das erste Byte des Programms selbst, so daß Sie die ausgegebenen Bytes mit der Programm-Liste vergleichen können.

Die Aussendung beginnt mit dem Start-Ø-Bit: Der Dezimalpunkt verlischt. Anschließend wird das Bit Nr. Ø mit dem Wert 1 des Bytes CD (= 11ØØ 11Ø1) ausgesendet: Der Dezimalpunkt leuchtet. Es folgt wieder ein Ø-Bit (Dezimalpunkt verlischt) und dann folgen zwei 1-Bits, während denen die Anzeige ununterbrochen leuchtet. Auf diese Weise werden alle acht Daten-Bits bis zum Bit Nr. 7, das den Wert 1 hat, ausgesendet. Als Abschluß der Aussendung des ersten Bytes kommen die beiden Stop-1-Bits.

Rein optisch werden Sie kaum in der Lage sein, die ausgesendeten Bits auf der Datenleitung zu identifizieren, da sich der Signal-Pegel z. B. bei einer Reihe von 1-Bits nicht ändert. Es ist dann nicht auszumachen, ob der unveränderte Signal-Zustand zwei oder drei gleiche Bits signalisiert.

Wir haben in das Programm eine akustische Hilfe eingebaut: Das Start-0-Bit wird durch einen tiefen Ton markiert, jedes der Daten-Bits durch einen Ton mittlerer Frequenz und die beiden Stop-Bits durch je einen hohen Ton.

Diese Ton-Aussendung hat mit der eigentlichen, Bit-seriellen Ausgabe natürlich nichts zu tun. Sie dient rein der Demonstration des Prinzips.

Dieses Prinzip können Sie noch deutlicher erkennen, wenn Sie im Programm die angegebenen NOP-Befehle wie folgt ändern:

Adresse	Bytes	Befehl
184C	CD 8C 18	CALL AUSGAB
185B	CD 8C 18	CALL AUSGAB

Starten Sie das Programm und sagen Sie dem System „Clear To Send" (CTS mit der Taste USER KEY): Du darfst senden!

Die den ausgesandten Signalen entsprechenden Bit-Werte werden nun in der Anzeige sichtbar. Leider hat die Anzeige des Micro-Professors nur 6 Stellen. Die beiden höchstwertigen Bits können also nicht angezeigt werden. Sie erkennen aber, was gemeint ist.

Wohlgemerkt: Auch diese Anzeige ist zwar recht hübsch; sie hat aber mit der eigentlichen, seriellen Ausgabe nichts zu tun.

Das Aussenden von Daten wird nur solange fortgesetzt, wie die empfangende Peripherie ein CTS-Signal an den Sender schickt. Wird das CTS-Signal weggenommen, dann wird die begonnene Aussendung eines Bytes zu Ende geführt und dann die Aussendung solange unterbrochen, bis das CTS-Signal wieder erscheint. Diese Eigenschaft unseres Programms entspricht den tatsächlichen Verhältnissen bei der seriellen Daten-Ausgabe.

Nun werden Sie mit einigem Recht fragen, was dieser Versuch wohl mit einem Interface zu tun hat, um das es ja eigentlich geht. – Sie werden es gleich sehen.

Das Hauptprogramm des gerade beschriebenen Versuchs und die Unterprogramme CTS und SEROUT können in der angegebenen Form auch für eine praktisch verwertbare serielle Daten-Ausgabe verwendet werden. Für die praktische Verwendung müßten die im Unterprogramm SEROUT aufgerufenen Routinen STARTB, HIAUS und LOAUS sowie STOPB geändert werden. Mit diesen Routinen werden in unserem Programm die ausgegebenen Signalwerte des Start-Bits, der 1- und Ø-Daten-Bits und der Stop-Bits optisch und akustisch angezeigt.

Sehen Sie sich diese Routinen in der Auflistung unseres Programms an! Sie erkennen, daß in all diesen Routinen ein Modul WARTE aufgerufen wird. Dieses Modul sorgt dafür, daß das ausgegebene Signal für die Dauer einer Bit-Zeit unverändert erhalten bleibt.

Wie auch immer die Routinen STARTB, HIAUS, LOAUS und STOPB in einem praktisch anwendbaren Programm aussehen mögen: Ein Modul WARTE muß in jedem Fall eingebaut werden, damit die Ausgangssignale für eine definierte Zeit unverändert bleiben.

Ein WARTE-Modul läßt sich einfach programmieren, indem man den Mikroprozessor in eine Programm-Schleife schickt, in der er nichts anderes tut, als einen voreingestellten Zähler zu dekrementieren und jeweils nachzusehen, ob dieser Zähler bereits „leergezählt" worden ist. Die Schleife wird erst dann verlassen, wenn der Zähler das Byte ØØ enthält. Der anfangs in den Zähler gesetzt Wert bestimmt die Verweildauer des Programms in der WARTE-Schleife.

In unserem Programm nutzen wir die Verweildauer des Programms zur Generierung der Töne, die aber in einem praktisch verwertbaren Programm nichts zu tun haben. Mit anderen Worten: In einem solchen

Programm wird der Mikroprozessor für die gesamte Dauer der Daten-Ausgabe einfach lahmgelegt. Er verbringt die meiste Zeit in WARTE-Schleifen – und könnte doch sicher während dieser Zeit sehr viel nützlichere Aufgaben erledigen.

Selbst dann, wenn es mit irgendeinem Trick gelingt, die WARTE-Zeiten sinnvoll auszunutzen, hat die CPU bei der seriellen Ausgabe von Bytes noch alle Hände voll zu tun, die auszugebenden Bytes der Peripherie bitweise zur Verfügung zu stellen. Außerdem muß sie am Anfang eines jeden Bytes ein Start-Bit und am Ende eines jeden Bytes Stop-Bits generieren.

Ideal wäre es, wenn die CPU diese Aufgabe an eine Anordnung delegieren könnte, der einfach ein Byte übergeben wird und die dann alle weiteren Tätigkeiten für die serielle Ausgabe übernimmt.

Genau eine solche Anordnung bezeichnet man als **Interface.** (Sprich: interfäiss.)

Sehen Sie sich bitte das Bild H7.1 an und vergleichen Sie es mit dem Bild H2.1! In beiden Fällen soll ein Mikroprozessor-System mit vier Peripherie-Einheiten Daten austauschen. Es wird zunächst nichts darüber ausgesagt, in welcher Form die Peripherie-Einheiten die Daten zur Verfügung gestellt haben wollen.

Wir wollen willkürlich annehmen, der Drucker setze eine serielle **Schnittstelle** voraus, er sei also darauf eingerichtet, daß ihm Daten Bit-seriell, Byte-seriell vom System angeliefert werden.

In der Anordnung entsprechend dem Bild H2.1 gibt es keine andere Möglichkeit, als die CPU so zu programmieren, wie wir es eben beschrieben haben. Die CPU muß dem Drucker die Daten also bitweise z.B. über eine Leitung des Datenbus anliefern. Verabredungsgemäß muß sie jeweils vor den Daten-Bits ein Start-0-Bit und nach den Daten-Bits zwei Stop-1-Bits aussenden, und das mit einer genau definierten Geschwindigkeit, denn so und nicht anders will es der Drucker haben.

Ganz anders sieht die Sache bei der Anordnung entsprechend dem Bild H7.1 aus. Die bitweise Aufbereitung der Daten in die vom Drucker gewünschte Form übernimmt hier das Interface.

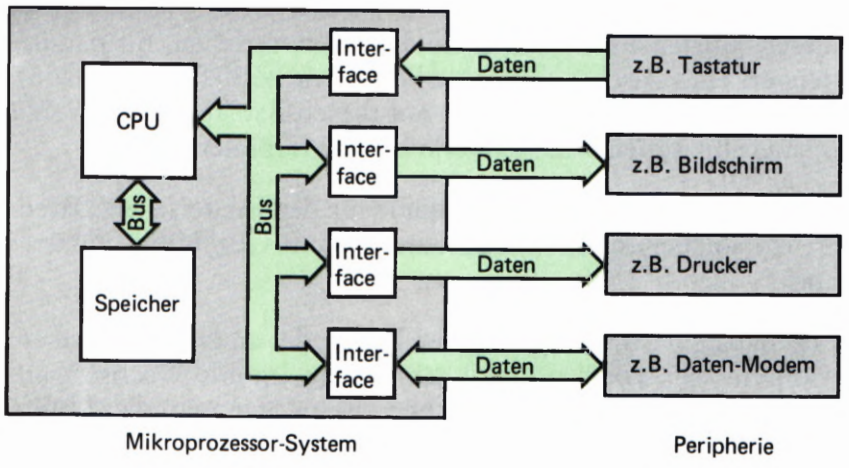


Bild H7.1
In einem Mikroprozessor-System übernimmt das Interface die Aufbereitung der Daten in eine Form, wie sie für die jeweilige Peripherie gebraucht wird.

Das Interface ist eine Anordnung, der nach Art des I/O-Mapping (vgl. Lehrgang Mikroprozessortechnik, Seite H61) eine normale Adresse zugeteilt werden kann. Die CPU behandelt das Interface dann gerade so, als wäre es eine Speicherzelle. Sie übergibt das auszusendende Daten-Byte z. B. mit einem LD (adr),A-Befehl dem Interface; mit dem Aussenden und dem Generieren von Start- und Stop-Bits hat sie nichts mehr zu tun. Das ist Sache des Interface.

Üblicherweise wird der Datenverkehr zwischen CPU und Interface in einem Isolated-I/O-System vorgenommen (vgl. Lehrgang Mikroprozessortechnik, Seite H64). Die CPU liefert die auszusenden Daten an das Interface über eine Port-Adresse, z. B. mit dem Befehl OUT (adr),A. Auch jetzt übernimmt das Interface die Aufbereitung der Daten zur seriellen Aussendung.

Die CPU darf dem Interface immer erst dann das nächstfolgende Daten-Byte anliefern, wenn das Interface mit der Aufbereitung des vorhergehenden Daten-Bytes fertig ist. Die Bereitschaft, ein neues Daten-Byte zur Aussendung aufbereiten zu können, teilt das Interface der CPU mit einem speziellen Signal mit. Dieses Signal löst bei der CPU einen **Interrupt** aus (vgl. Lehrgang Mikroprozessortechnik, Seite S9), mit dem das nächstfolgende Daten-Byte von der CPU angefordert wird.

Am hier beschriebenen Beispiel erkennen Sie, daß ein Interface eine Anordnung ist, die sonst von der CPU zu lösende Aufgaben übernimmt. Es hat also selbst gewisse Ähnlichkeiten mit einer CPU. Tatsächlich ist ein Interface ein Rechner, der auf die Lösung einer ganz bestimmten Aufgabe zugeschnitten ist. Weil diese Aufgabe unterschiedlich gestellt werden kann (z. B. die serielle Aussendung von Daten mit wählbarer Geschwindigkeit), muß das Interface – ähnlich wie eine CPU – programmierbar sein. Wie das gemacht wird, wollen wir Ihnen in diesem Lehrgang zeigen.

Bei der Anwendung von Mikroprozessoren fallen immer wiederkehrende Interface-Aufgaben an. Es gibt deshalb zu jeder Mikroprozessor-Familie Standard-Interfaces (sprich: interfäissis), die die gesamte zugehörige Elektronik in einem einzigen IC vereinigen. Man spricht dabei von **Interface-Bausteinen.**

Wir werden Ihnen in diesem Lehrgang zwei Interface-Bausteine der Z 80-Familie und einen weiteren Interface-Baustein der 8085-Familie vorstellen.

Zwei dieser Bausteine dienen speziell dem Byte-seriellen, Bit-parallelen Datenverkehr zwischen CPU und Peripherie (Z 80 PIO und 8255). Wir werden Ihnen zeigen, wie man mit diesen Bausteinen auch sehr elegant einen Bit-seriellen Datenverkehr programmieren kann.

Es gibt auch spezielle Interface-Bausteine für den Bit-seriellen Datenverkehr; ein solcher Interface-Baustein ist nicht Gegenstand unseres Lehrgangs.

Der dritte Baustein ist ein sogenannter **Timer,** der an die CPU oder an die Peripherie vier voneinander unabhängige Impuls-Wechselspannungen liefern kann. Wir werden Ihnen zeigen, wie man diese Möglichkeiten vorteilhaft nutzen kann.

Die Peripherie-Leiterplatte

Die Kenntnisse über die Arbeitsweise und die Programmierung von Interface-Bausteinen – die auch als **Schnittstellen-Bausteine** bezeichnet werden – wollen wir Ihnen anhand von Versuchen mit dem Micro-Professor vermitteln.

Der Micro-Professor enthält in der vom Hersteller ausgelieferten Form bereits einen Schnittstellen-Baustein, der allerdings innerhalb des Systems mit der Abfrage der Tastatur und der Steuerung der Sieben-Segment-Anzeige voll ausgelastet ist. Für die Bedienung anderer Peripherie steht dieser Baustein nicht zur Verfügung. Es handelt sich um den Baustein 8255. (Vgl. Lehrgang Mikroprozessortechnik, Seite T 1).

Für den Anschluß externer Peripherie hat der Hersteller auf der System-Leiterplatte zwei zunächst unbestückte Fassungen vorgesehen, in die Peripherie-Bausteine nachträglich eingesteckt werden können. Diese beiden Bausteine finden Sie in der Material-Lieferung zu Ihrem Lehrgang.

Die Peripherie-Anschlüsse der Bausteine sind auf der Micro-Professor-Leiterplatte an eine 40polige Stiftleiste geführt, die mit PIO CTC I/O BUS bezeichnet ist.

Die Bedienung externer Peripherie kann naturgemäß nur dann experimentell erläutert werden, wenn auch wirklich eine Peripherie vorhanden ist. Da solche Peripherie sehr unterschiedlich aussehen kann (vgl. Bild H 2.1), liefern wir Ihnen mit diesem Lehrbrief eine Peripherie-Leiterplatte, auf der zwar keine „echte" Peripherie angeordnet ist, auf der aber mit Tasten und Anzeigen eine beliebige Peripherie simuliert werden kann.

Zusätzlich zu dieser Simulations-Peripherie ist auf der Peripherie-Leiterplatte noch einmal der gleiche Schnittstellen-Bausteine 8255 angeordnet, der bereits in der Grund-Version des Micro-Professors vorhanden ist. Dieser Baustein 8255 steht für unsere Versuche völlig frei zur Verfügung.

Damit Ihre Peripherie-Leiterplatte auch über die Versuche des Lehrgangs hinaus verwendet werden kann, sind einige Peripherie-Anschlüsse der Schnittstellen-Bausteine an eine 15polige Klemmleiste geführt. Hier kann bei Bedarf „echte" Peripherie angeschlossen werden. Das Bild H 9.1 zeigt die Belegung dieser Klemmleiste. Was es mit den eingetragenen Bezeichnungen auf sich hat, wird im Laufe des Lehrgangs erläutert.

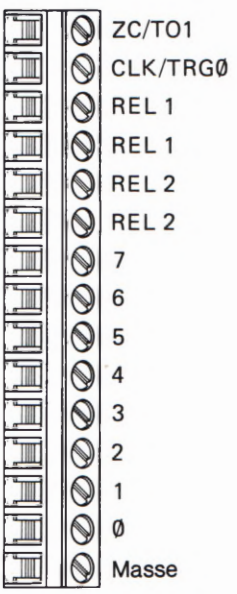


Bild H 9.1
Über die Klemmleiste auf der Interface-Leiterplatte kann „echte" Peripherie angeschlossen werden.

Z 80 PIO und Z 80 CTC im Micro-Professor

Im Bild H 10.1 sehen Sie einen Ausschnitt der Leiterplatte des Micro-Professors. Orientieren Sie sich bitte, welchen Ausschnitt wir dargestellt haben. (Vgl. auch Lehrgang Mikroprozessortechnik, Seite T 1).

In Ihrer Material-Lieferung finden Sie die beiden ICs Z 80 PIO (Teilenummer 31 110) und Z 80 CTC (Teilenummer 31 111). – **PIO** ist die

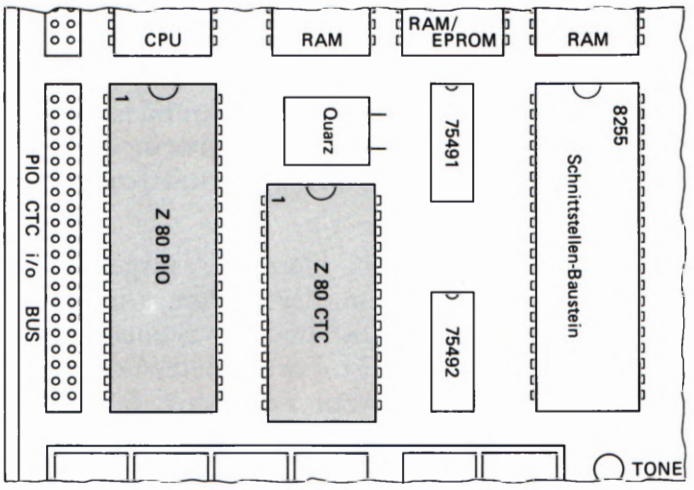


Bild H 10.1
Orientieren Sie sich auf der Leiterplatte des Micro-Professors, wo die Fassungen für die Z 80 PIO und den Z 80 CTC liegen. Achten Sie sorgfältig auf die richtige Anordnung der ICs.

Abkürzung für **P***arallel* **I***nput/***O***utput Controller* (Steuer-Baustein für parallele Ein- und Ausgabe). Es handelt sich also um einen Schnittstellen-Baustein für Bit-parallele, Byte-serielle Datenübertragung (vgl. Seite H 3). Mit diesem Baustein werden wir uns in diesem Lehrgang beschäftigen.

CTC ist die Abkürzung für **C***ounter/***T***imer* **C***ircuit* (Zähler/Zeitgeber Schaltung). Wir stellen Ihnen den Z 80 CTC in einem der folgenden Lehrbriefe vor.

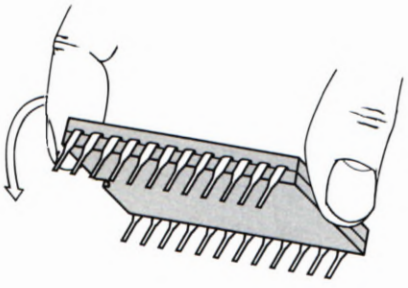


Bild H 10.2
So können die IC-Anschlüsse senkrecht zum IC-Gehäuse ausgerichtet werden.

Setzen Sie die beiden ICs in die entsprechenden, im Bild H 10.1 durch graue Unterlegungen markierten Fassungen. Achten Sie sorgfältig darauf, daß bei beiden ICs die Markierung an der einen Schmalseite des ICs für den Anschluß 1 so angeordnet ist, wie wir es im Bild dargestellt haben. Bei falscher Anordnung kann der nicht billige Baustein zerstört werden!

Vermutlich sind die „Beinchen" der ICs zunächst soweit gespreizt, daß sie nicht in die Kontakte der Fassungen passen. Biegen Sie in diesem Fall beide „Beinchen"-Reihen der ICs nacheinander vorsichtig nach innen, wie es im Bild H 10.2 dargestellt ist.

Anschluß der Peripherie-Leiterplatte an den Micro-Professor

Die Maße der Peripherie-Leiterplatte haben wir so gewählt, daß sie in der linken Hälfte des Buch-förmigen Gehäuses des Micro-Professors Platz findet. – Im Bild H 11.1 erkennen Sie, wie diese Leiterplatte an den Micro-Professor angeschlossen wird.

Der Anschluß wird über zwei 40polige Flachbandleitungen vorgenommen, die einseitig fest mit der Peripherie-Leiterplatte verbunden sind.

Der freie Stecker der Ihnen zugewandten Flachbandleitung wird auf die mit Z 80 CTC I/O BUS bezeichnete, 40polige Stiftleiste des Micro-Professors gesteckt. Achten Sie darauf, daß beim Stecken auch wirklich alle 40 Stifte der Stiftleiste erfaßt werden! – Die Flachbandleitung verbindet die Peripherie-Anschlüsse der Bausteine Z 80 PIO und Z 80 CTC mit der Simulations-Peripherie.

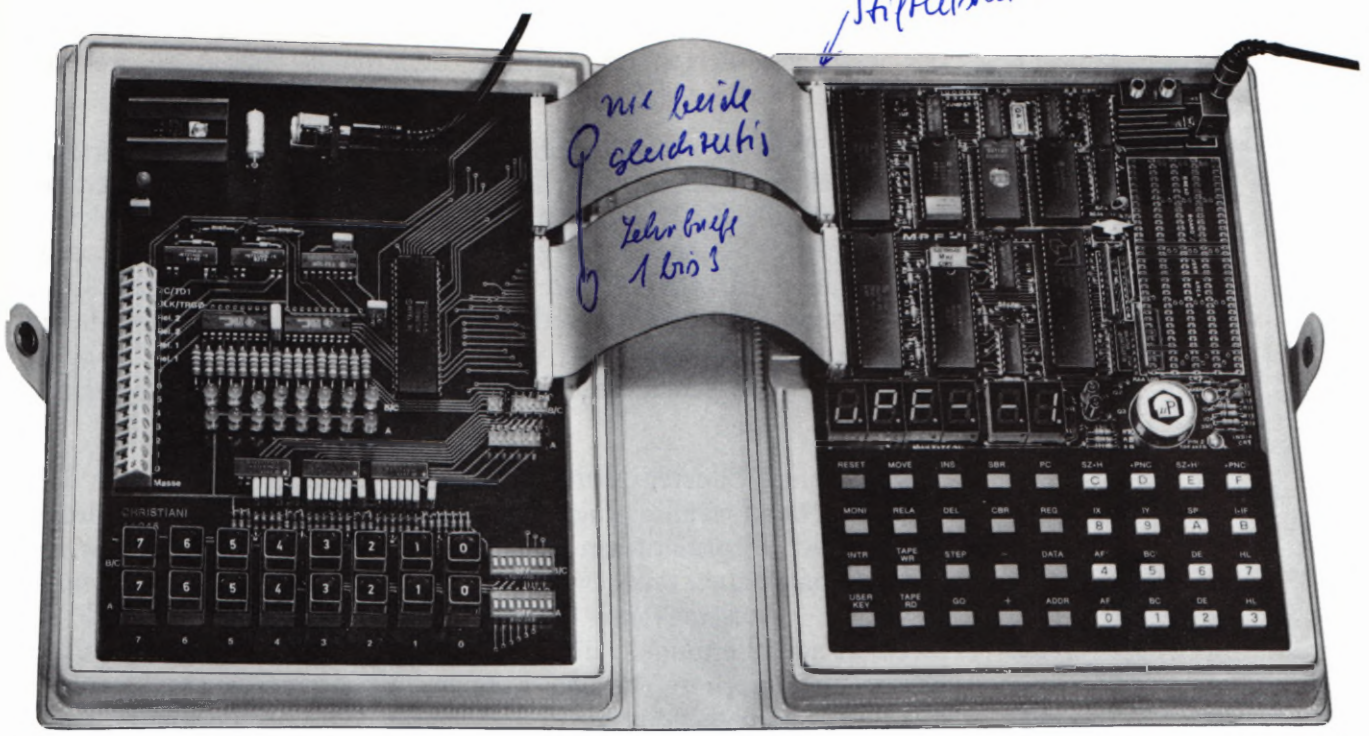


Bild H 11.1
Die Peripherie-Leiterplatte wird über zwei 40polige Flachbandleitungen mit dem Micro-Professor verbunden.

Der freie Stecker der von Ihnen abgewandten Flachbandleitung wird auf die mit Z 80 CPU BUS bezeichnete, 40polige Stiftleiste des Micro-Professors gesteckt. Mit dieser Flachbandleitung wird der auf der Peripherie-Leiterplatte untergebrachte Baustein 8255 von der CPU her mit Adressen, Daten und Steuer-Signalen versorgt. Bei den Versuchen mit der Z 80 PIO und dem Z 80 CTC tritt diese Verbindung nicht in Funktion.

Die Peripherie-Leiterplatte wird über ein eigenes Netzgerät mit Spannung versorgt. Dieses Netzgerät entspricht dem des Micro-Professors: Es wird in die Netzsteckdose gesteckt und liefert über seine Ausgangsleitung eine unstabilisierte Spannung von etwa 9V. Auf der Peripherie-Leiterplatte ist hinter der Anschlußbuchse eine Stabilisierungsschaltung angeordnet, welche die Betriebsspannung von 5V zur Verfügung stellt.

Die Funktion der Peripherie-Leiterplatte

Mit Hilfe der auf der Peripherie-Leiterplatte untergebrachten Bauelemente wollen wir Ihnen nacheinander die Arbeitsweise von drei Peripherie-Bausteinen zeigen. Da wir uns in diesem Lehrbrief zunächst mit der Z 80 PIO beschäftigen wollen, wird hier zunächst nur der Teil der Schaltung betrachtet, der mit der Z 80 PIO zu tun hat. Die übrigen Funktionen der Schaltung lernen Sie dann im Zusammenhang mit den anderen Schnittstellen-Bausteinen kennen.

Sehen Sie sich bitte noch einmal das Bild H 7.1 an! Es handelt sich um eine Prinzip-Darstellung, bei der auf alle Einzelheiten verzichtet wurde. Sie erkennen aber, daß jeder der angedeuteten Peripherien ein

H
12

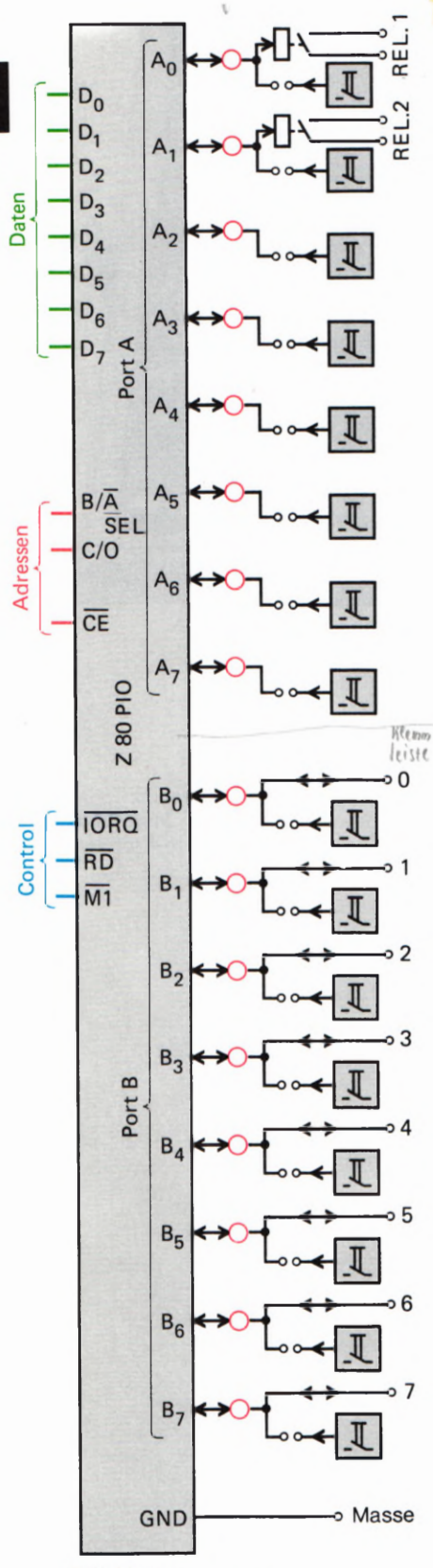


Bild H12.1
Jedem Port-Anschluß auf der Peripherie-Leiterplatte ist eine Leuchtdiode zugeordnet. Die Tasten werden über steckbare Brücken aktiviert.

eigenes Interface zugeordnet wird. Aufgabe des Interface ist es, die von der jeweiligen Peripherie verlangte Schnittstelle bereit zu stellen. Genau das ist die Aufgabe eines Schnittstellen-Bausteins.

Die Z80 PIO ist ein Baustein, der eine Bit-parallele, Byte-serielle Schnittstelle realisieren kann. Diese Aussage ist allerdings nur gerade halb richtig: Die Z80 PIO kann nämlich nicht nur eine, sondern gleich zwei derartige Schnittstellen realisieren. Wenn man voraussetzt, daß die im Bild H7.1 angedeuteten vier Peripherien sämtlich Bit-parallele, Byte-serielle Schnittstellen verlangen, dann kann der Datenverkehr des Mikroprozessor-Systems über zwei Bausteine Z80 PIO abgewickelt werden: Jeder dieser Bausteine enthält zwei Interface-Einheiten.

Dem Interface-Baustein werden die Daten für die Peripherie von der CPU über den Datenbus zugestellt. Daß der Baustein zu seiner Adressierung auch Informationen über den Adreßbus und zusätzlich noch Steuersignale haben muß, braucht an dieser Stelle noch nicht zu interessieren. – Ausgangsseitig hat der Schnittstellen-Baustein Z80 PIO zweimal acht Leitungen, über die jeweils die acht Bits eines Bytes parallel transportiert werden können: Wahlweise entweder vom Mikroprozessor-System zur Peripherie oder von der Peripherie zum Mikroprozessor-System.

Sehen Sie sich jetzt bitte das Bild H12.1 an! Zunächst ohne Rücksicht darauf, auf welcher Leiterplatte die einzelnen Funktionen untergebracht sind, haben wir den Schnittstellen-Baustein Z80 PIO dargestellt. Links oben ist grün der Anschluß an den Datenbus des Systems eingetragen. Darunter erkennen Sie die zunächst noch nicht interessierenden Adreß- und Steuer- (Control-)Leitungen.

Rechts sind deutlich die zweimal acht Anschlüsse für jeweils acht Bits eines Bytes zu sehen, an die insgesamt zwei Peripherien angeschlossen werden können. Jede Gruppe von acht Anschlüssen wird als **Port** bezeichnet. Oben liegt der Port A mit den Anschlüssen AØ bis A7, unten der Port B mit den Anschlüssen BØ bis B7.

In unserer Darstellung finden Sie an jedem Port-Anschluß einen kleinen, roten Kreis, der eine Leuchtdiode symbolisiert. Diese Darstellung ist keineswegs DIN-gerecht. Wir wollen damit lediglich andeuten, daß die Leuchtdiode das am jeweiligen Anschluß liegende Signal anzeigt, gleichgültig, ob es vom Mikroprozessor-System als Sender an die empfangende Peripherie geschickt wird, oder ob die Peripherie das Signal sendet und das Mikroprozessor-System der Empfänger ist.

Das Bild H12.1 zeigt, daß auf der Peripherie-Leiterplatte die beiden Ports der Z80 PIO unterschiedlich beschaltet sind. Sehen Sie sich zunächst den Port B an!

Die Leuchtdioden zeigen die an den Port-Anschlüssen BØ bis B7 liegenden Signale an, gleichgültig, ob diese Signale von der Z80 PIO gesendet oder empfangen werden. – Die Port-Anschlüsse werden dann an die Anschlüsse 0 (BØ) bis 7 (B7) der Klemmleiste geführt. Dort kann bei Bedarf eine „echte" Peripherie angeschlossen werden.

In unseren Versuchen werden wir uns begnügen, die vom Mikroprozessor-System über den Schnittstellen-Baustein Z80 PIO an die Peripherie gesendeten Signale von der Leuchtdiode anzeigen zu lassen.

Die im Bild als kleine Kästchen dargestellten Schalter auf der Peripherie-Leiterplatte erlauben es, von einer Peripherie an das System gesendete Signale zu imitieren. Ein betätigter Schalter liefert ein 1-Signal (+ 5V) an den entsprechenden Port-Anschluß.

Ganz offensichtlich besteht ein grundsätzlicher Unterschied darin, ob ein Port-Anschluß der Z 80 PIO 0- oder 1-Signale (0V oder + 5V) an die Peripherie sendet, oder ob er von der Peripherie solche Signale empfangen kann. Wir werden Ihnen in den folgenden Abschnitten zeigen, wie die Z 80 PIO durch entsprechende Programmierung intern auf sendende oder empfangende Port-Anschlüsse umgeschaltet werden kann. Jedenfalls ist eins deutlich: Wenn ein Port-Anschluß auf Senden programmiert wurde, dann darf es auf keinen Fall passieren, daß er gleichzeitig über den zugehörigen Schalter mit einem äußeren Signal beaufschlagt wird. Die Schalter auf der Peripherie-Leiterplatte können deshalb durch steckbare Brücken (Steck-Bügel) nach Bedarf wirksam oder unwirksam gemacht werden.

Der Port A ist über die Peripherie-Leiterplatte ähnlich beschaltet wie der Port B. Hier wurde jedoch auf die Durchschaltung der Port-Anschlüsse an die Klemmleiste verzichtet.

Die Port-Anschlüsse A0 und A1 sind dafür an je ein Dual-in-Line-Relais geführt, dessen geschaltete Kontakte an den mit Rel. 1 bzw. Rel. 2 gekennzeichneten Anschlüssen der Klemmleiste liegen. So ist es möglich, zwei Peripherie-Leitungen über die als Sender programmierten Port-Anschlüsse A0 und A1 potentialfrei zu schalten.

Wo die im Bild H 12.1 dargestellten Funktionen tatsächlich untergebracht sind, bedarf kaum einer Erläuterung: Die Z 80 PIO befindet sich auf der Leiterplatte des Micro-Professors. Ihre Port-Leitungen werden über die 40polige Stiftleiste PIO CTC I/O BUS und die 40polige Flachbandleitung auf die Peripherie-Leiterplatte geführt. Dort sind die Leuchtdioden, die Schalter und die zugehörigen Steck-Bügel sowie die Klemmleiste angeordnet.

Einen Schaltplan der Funktionen auf der Peripherie-Leiterplatte zeigen wir im Bild H 13.1. Die Anordnung der Schalter mit dem nachfolgenden Schmitt-Trigger sowie die Anzeige-Leuchtdiode mit ihrer Ansteuerung ist in der dargestellten Form insgesamt 16 mal vorhanden. – Die gestrichelt eingetragene Verbindung zur Klemmleiste gilt für die Anschlüsse des Ports A. Das Relais mit seinem Steuer-Tran-

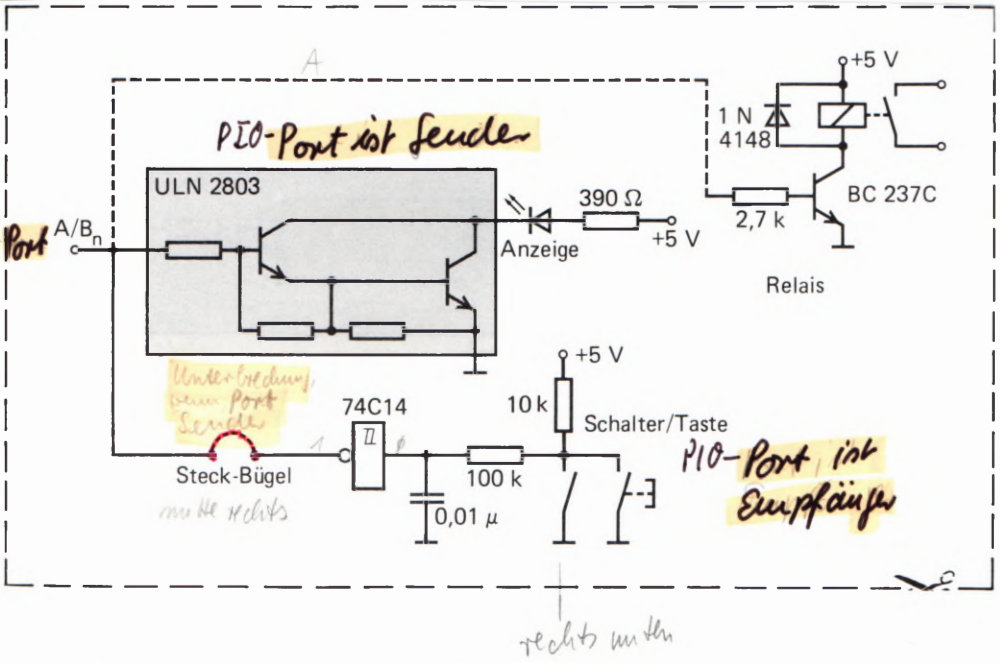


Bild H 13.1
Diese Schaltung ist auf der Peripherie-Leiterplatte sechzehnmal vorhanden. Der gestrichelt dargestellte Relais-Anschluß gilt nur für die Anschlüsse 0 und 1 des Ports A.

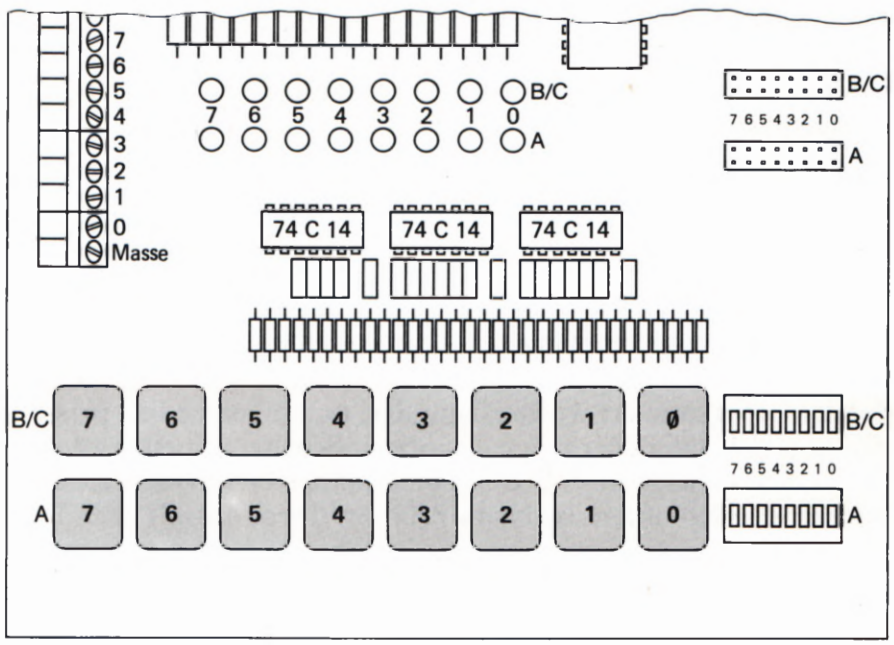


Bild H 14.1
Der hier dargestellte Teil der Peripherie-Leiterplatte ist für Eingaben und Ausgaben von Daten zuständig.

sistor ist in der Schaltung nur für die Port-Anschlüsse A0 und A1 realisiert.

Der Schaltplan zeigt, daß an jedem Port-Anschluß zwei Schalter liegen. Der eine ist ein Tast-Schalter, der nur solange als 1-Signal an den entsprechenden Port-Anschluß liefert, wie er betätigt ist. Der zweite Schalter ist jeweils in einer 8-fach Dual-in-Line-Anordnung untergebracht. Dieser Schalter erlaubt es, ein Dauer-1-Signal an den zugehörigen Port-Anschluß zu legen.

Das den Schaltern folgende RC-Glied fängt das unvermeidliche Prellen der Schalter auf. Der dadurch entstehende, langsame Signalwechsel wird vom nachgeschalteten Schmitt-Trigger in eine steile Schalt-Flanke gewandelt.

Die der Anzeige-Leuchtdiode vorgeschaltete Transistor-Anordnung ist Teil einer integrierten Schaltung (ULN 2004). Ohne diese Transistor-Anordnung würde die Leuchtdiode den Port-Anschluß der Z 80 PIO in unzulässiger Weise belasten.

Wir haben Ihnen hier nur den Teil der Schaltung der Interface-Leiterplatte vorgestellt, der im Zusammenhang mit der Z 80 PIO von Interesse ist. Die übrige Schaltung erläutern wir zusammen mit der Besprechung der weiteren Interface-Bausteine.

Das Bild H 14.1 zeigt die Anordnung der hier vorgestellten Funktionen auf der Interface-Leiterplatte.

Die Adressierung der Z 80 PIO

Aufgabe eines Schnittstellen-Bausteins ist es, den Datenverkehr zwischen der CPU und einer an das Mikroprozessor-System angeschlossenen Peripherie zu vermitteln. Der Schnittstellen-Baustein muß also alle Daten, die ihm von der CPU oder von der Peripherie übergeben werden, baldmöglichst an den jeweils anderen Partner der Kommunikation weiterreichen.

Weil komfortable Schnittstellen-Bausteine die Daten-Übermittlung auf vielfältige Weise vornehmen können, müssen sie jeweils vor dem Beginn ihrer eigentlichen Funktion auf eine bestimmte Art der Daten-Übermittlung programmiert werden. Die Programmierung geschieht – ganz ähnlich wie bei einer CPU – durch Befehle in Form von Bytes, also ebenfalls durch Daten. Diese Befehls-Daten soll der Schnittstellen-Baustein aber nicht weiterreichen; er muß sie intern verarbeiten.

Zur Unterscheidung zwischen Daten-Bytes, die weitergereicht werden müssen, und Befehls-Bytes zur Programmierung muß der Schnittstellen-Baustein eine besondere Einrichtung besitzen, über die ihm mitgeteilt werden kann, ob er die ihm gerade von der CPU zugespielten Daten an die Peripherie weiterreichen oder ob er sie zu seiner eigenen Programmierung verwenden soll. (Vgl. Seite S 2.)

Häufig tritt das hier angedeutete Problem bei Schnittstellen-Bausteinen in genau gleicher Form sogar mehrfach auf. Wenn an ein und demselben Schnittstellen-Baustein mehr als ein Peripherie-Baustein angeschlossen werden kann, wenn er also mehr als einen Port für die Peripherie zur Verfügung hat, dann muß seine Kommunikations-Art für jeden dieser Ports getrennt programmiert werden (vgl. die Seiten S 2 und S 3). Außerdem muß der Schnittstellen-Baustein natürlich wissen, für welchen seiner Ports ein ihm zur Aussendung übergebenes Daten-Byte bestimmt ist. Es muß also eine weitere Einrichtung vorgesehen sein, mit welcher der Schnittstellen-Baustein feststellen kann, für welchen seiner Ports das ihm von der CPU übergebene Byte bestimmt ist.

Im Bild H 15.1 haben wir die Anschluß-Belegung des Z 80 PIO-ICs dargestellt. Zusätzlich sind die Funktionen der einzelnen Anschlüsse angedeutet. Ins Auge fallen sofort die beiden Gruppen für die Ports A und B mit jeweils 8 Anschlüssen. An die Anschlüsse 16 und 18 sind die Handshake-Leitungen des Ports A geführt; die Handshake-Leitungen für den Port B liegen an den Anschlüssen 17 und 21. (READY = ARDY, STROBE = ASTB jeweils für Port A. Beim Port B heißen die entsprechenden Anschlüsse BRDY bzw. BSTB. Vgl. Bild S 3.1.)

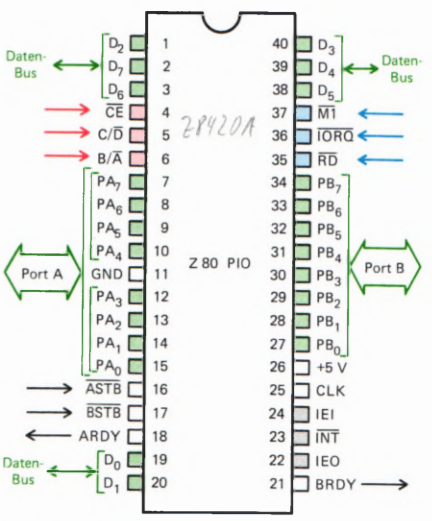


Bild H 15.1
Anschluß-Belegung der Z 80 PIO.

Wir wollen uns hier für die mit C/$\overline{D}$ (Anschluß 5) und mit B/$\overline{A}$ (Anschluß 6) bezeichneten Anschlüsse interessieren. Wenn Sie den Lehrgang Mikroprozessortechnik durchgearbeitet haben, dann erinnern Sie sich vielleicht, daß wir dort die Adreßleitungen rot markiert haben. (Vgl. z. B. die Bilder H 62.1 und H 63.1 im Lehrgang Mikroprozessortechnik.) Anscheinend haben also die Anschlüsse C/$\overline{D}$ und B/$\overline{A}$ – ebenso wie der benachbarte Anschluß $\overline{CE}$ – irgendetwas mit Adressen zu tun.

Zunächst wollen wir nur feststellen, daß der Anschluß C/$\overline{D}$ genau die Einrichtung zur Verfügung stellt, mit welcher der Schnittstellen-Baustein unterscheiden kann, ob ein ihm von der CPU übermitteltes Byte an die Peripherie weitergeschickt werden soll, oder ob dieses Byte zu seiner eigenen Programmierung dient.

Der Anschluß C/$\overline{D}$ (**C***ontrol*/**D***ata Select* – Steuerzeichen/Daten Auswahl) bestimmt die Verwendung eines der Z 80 PIO von der CPU übermittelten Bytes.

Wenn am Anschluß C/$\overline{D}$ ein 1-Signal liegt, dann interpretiert die Z 80 PIO ein ihr gleichzeitig von der CPU übermitteltes Byte als Steuerzeichen (Befehl) zur eigenen Programmierung. Die Programmierung bezieht sich auf den jeweils über den B/$\overline{A}$-Anschluß angewählten Port. (Vgl. die folgende Beschreibung des B/$\overline{A}$-Anschlusses.)

Wenn am Anschluß C/$\overline{D}$ ein 0-Signal liegt, dann wird das gleichzeitig von der CPU übermittelte Byte von der Z 80 PIO als Daten-Byte erkannt und an die Peripherie weitergeleitet.

Diese Beschreibung zeigt, daß das an den C/$\overline{D}$-Anschluß gelieferte Signal für einen Programmier-Befehl *active HIGH* ist: Ein 1-Signal kennzeichnet einen Befehl. Für ein Daten-Byte ist das Signal *active LOW*. Es wird mit einem 0-Signal gekennzeichnet. Die Überstreichung des Buchstabens D macht diese Tatsache deutlich.

Die zweite Einrichtung muß dem Schnittstellen-Baustein die Unterscheidung ermöglichen, ob ein Befehls-Byte den Port A oder den Port B betrifft. Diese Einrichtung bietet der Anschluß B/$\overline{A}$:

Der Anschluß B/$\overline{A}$ (Port **B**/Port **A**) bestimmt die Verwendung eines dem Schnittstellen-Baustein von der CPU übermittelten Bytes für den Port A bzw. für den Port B.

Wenn am Anschluß B/$\overline{A}$ ein 1-Signal liegt, dann ist das von der CPU an die Z 80 PIO übermittelte Byte für den Port B bestimmt. Es ist dabei vom Wert des Signals am Anschluß C/$\overline{D}$ abhängig, ob dieses Byte für den Port B ein Befehls-Byte oder ein Daten-Byte für die Peripherie ist.

Wenn am Anschluß B/$\overline{A}$ ein 0-Signal liegt, dann erkennt die Z 80 PIO, daß dieses Byte für den Port A bestimmt ist.

Das an den Anschluß B/$\overline{A}$ gelieferte Signal ist *active HIGH* für den Port B und *active LOW* für den Port A. Die Überstreichung des Buchstabens A macht das deutlich.

Die hier vorgestellten Einrichtungen der Z 80 PIO sind offenbar recht zweckmäßig. Die Frage ist allerdings, auf welche Weise die CPU die entsprechenden Informationen über die Verwendung der auf dem Datenbus übermittelten Bytes an die PIO liefert.

Ehe diese Frage beantwortet wird, soll zunächst noch eine andere Frage gestellt werden: Wie wird überhaupt entschieden, ob ein von der CPU über den Datenbus ausgesendetes Byte für einen vom Schnittstellen-Baustein bedienten Peripherie-Port bestimmt ist? Die CPU sendet doch auch solche Bytes über den Datenbus, die z. B. für die Ablage in einer Speicherzelle bestimmt sind!

Die gleiche Frage gilt selbstverständlich auch für die umgekehrte Richtung des Daten-Transports: Woher weiß die CPU, daß ein Byte auf dem Datenbus nicht aus einer Speicherzelle, sondern von einem Schnittstellen-Baustein kommt? Vielleicht enthält das System sogar mehrere Schnittstellen-Bausteine (vgl. Bild H7.1 und Seite H12).

Hier handelt es sich offenbar um die Frage nach der Adressierung, die wir im Lehrgang Mikroprozessortechnik ab der Seite H57 ausführlich behandelt haben. Die CPU bestimmt selbst durch das auf den Adreßbus gesetzte Bitmuster, für welche der System-Einheiten (Speicher oder Interface-Ports, vgl. Bild H7.1) die Daten auf dem Datenbus bestimmt sind bzw. welche der System-Einheiten Daten für die CPU auf den Datenbus setzen soll.

Die Kommunikation der CPU mit dem Speicher ist in diesem Zusammenhang nicht interessant. – Für die Kommunikation mit der Peripherie hat die CPU zwei Möglichkeiten (vgl. Seite H8): Sie kann der Peripherie eine normale Zwei-Byte-Adresse (16 Bit, z.B. 1FA8H) wie einer Speicherzelle zuordnen (I/O-Mapping); sie kann die Peripherie aber auch über eine 1-Byte-Portadresse (8 Bit, z.B. 7AH) in einem Isolated-I/O-System ansprechen. Welche dieser Möglichkeiten gewählt wird, hängt von der Schaltung ab, mit welcher der Schnittstellen-Baustein angeschlossen ist.

Wir haben bereits erwähnt (Seite H8), daß der Datenverkehr zwischen der CPU und der Peripherie vorzugsweise in einem Isolated-I/O-System abgewickelt wird: Die CPU übergibt der Peripherie ihre Daten über den Schnittstellen-Baustein mit einem OUT-Befehl und holt Daten von der Peripherie über den Schnittstellen-Baustein mit einem IN-Befehl ab. Sie setzt dazu die 8 Bit der 1-Byte-Port-Adresse auf die Leitungen für die niederwertigen 8 Bit des Adreßbus (vgl. Lehrgang Mikroprozessortechnik, Bild H62.1b).

An dieser Stelle sei angemerkt, daß der Begriff Port doppeldeutig ist. Für die CPU ist ein Port der Zugang zu einem Etwas, das mit einem IN- oder OUT-Befehl angesprochen wird. Im hier betrachteten Fall ist das Etwas der Schnittstellen-Baustein, den die CPU mit einer 1-Byte-Port-Adresse anspricht. – Für den Schnittstellen-Baustein ist ein Port die Stelle, an der eine Peripherie angeschlossen wird. Die Daten-Kommunikation zwischen CPU und Peripherie wird also über zwei sozusagen hintereinander liegende Ports abgewickelt: Über den CPU-Port und über den Port des Schnittstellen-Bausteins.

Sehen Sie sich bitte das Bild H19.1 an! Sie erkennen die Z80 CPU mit den in diesem Zusammenhang interessierenden Anschlüssen im linken Teil des Bildes; rechts ist das Prinzip der Z80 PIO dargestellt. Die Daten-Transportleitungen (Datenbus) haben wir grün eingetragen und die zum Steuerbus gehörenden Leitungen blau. Wir interessieren uns hier für die rot eingetragenen Leitungen, die für die Adressierung zuständig sind.

Die CPU hat insgesamt 16 Anschlüsse für den Adreßbus, von denen in einem Isolated-I/O-System nur die acht Anschlüsse für die niederwertigen Adreß-Bits benötigt werden: Nur auf diese Anschlüsse schaltet die CPU eine Adreß-Information bei IN- und OUT-Befehlen.

Auffällig ist, daß die Z80 PIO offenbar mit nur vier der acht zu Verfügung stehenden Adreß-Bits adressiert wird. Man sagt: Die CPU-

Port-Adresse für die PIO ist nur unvollständig decodiert. Sie werden gleich sehen, was das bedeutet.

Zunächst wollen wir die möglichen Informationen, die der Z 80 PIO über ihre Anschlüsse C/$\overline{D}$ und B/$\overline{A}$ gegeben werden können, noch einmal übersichtlich zusammenstellen:

C/$\overline{D}$	B/$\overline{A}$	Sedez.	Information
Ø	Ø	Ø	Daten für Port A
Ø	1	1	Daten für Port B
1	Ø	2	Befehl für Port A
1	1	3	Befehl für Port B

Im Bild H 19.1 sehen Sie, daß in unserer Schaltung (die der im Micro-Professor realisierten Schaltung entspricht) der PIO-Anschluß B/$\overline{A}$ an den CPU-Anschluß für das Adreß-Bit Nr. Ø geführt ist und entsprechend der PIO-Anschluß C/$\overline{D}$ an den Anschluß für das Adreß-Bit Nr. 1.

Wenn wir die Beschreibung des noch nicht erwähnten PIO-Anschlusses $\overline{CE}$ mit den Adreß-Bits Nr. 6 und Nr. 7 zunächst außer Betracht lassen, dann ergibt sich für die Adressierung der PIO unter Berücksichtigung der verwendeten Adreß-Bits Nr. Ø und Nr. 1 folgendes Schema:

Bit						C/$\overline{D}$	B/$\overline{A}$	Sedez.	
7	6	5	4	3	2	1	Ø	Adr.	Information
X	X	X	X	X	X	Ø	Ø	Ø Ø	Daten für Port A
X	X	X	X	X	X	Ø	1	Ø 1	Daten für Port B
X	X	X	X	X	X	1	Ø	Ø 2	Befehl für Port A
X	X	X	X	X	X	1	1	Ø 3	Befehl für Port B

In diesem Schema bedeuten die Eintragungen X *Don't-care*-Werte, also beliebige Werte Ø oder 1.

Es werde zunächst angenommen, daß an den Stellen X jeweils Werte Ø stehen. Die CPU teilt dann jedesmal, wenn sie mit einem Befehl OUT (ØØ),A den Inhalt des Akkumulators über den Datenbus an ihre Port-Adresse ØØ schickt, gleichzeitig der Z 80 PIO mit, daß jetzt auf dem Datenbus solche Daten stehen, die über den PIO-Port A an die Peripherie weitervermittelt werden sollen. (Lesen Sie bitte noch einmal den Abschnitt über die IN- und OUT-Peripherie-Befehle des Z 80 ab der Seite H 63 im Lehrgang Mikroprozessortechnik nach!)

Mit dem Befehl OUT (Ø2),A setzt die CPU ebenfalls den Inhalt des Akkumulators auf den Datenbus. Die Port-Adresse Ø2 teilt aber diesmal der Z 80 PIO mit, daß es sich bei dem Byte auf dem Datenbus diesmal um ein Befehls-Byte zur Programmierung des Ports A handelt.

Aufgabe H 18.1

Was bewirkt die Befehlsfolge

```
LD    B,6FH
LD    C,Ø3H
OUT   (C),B
```

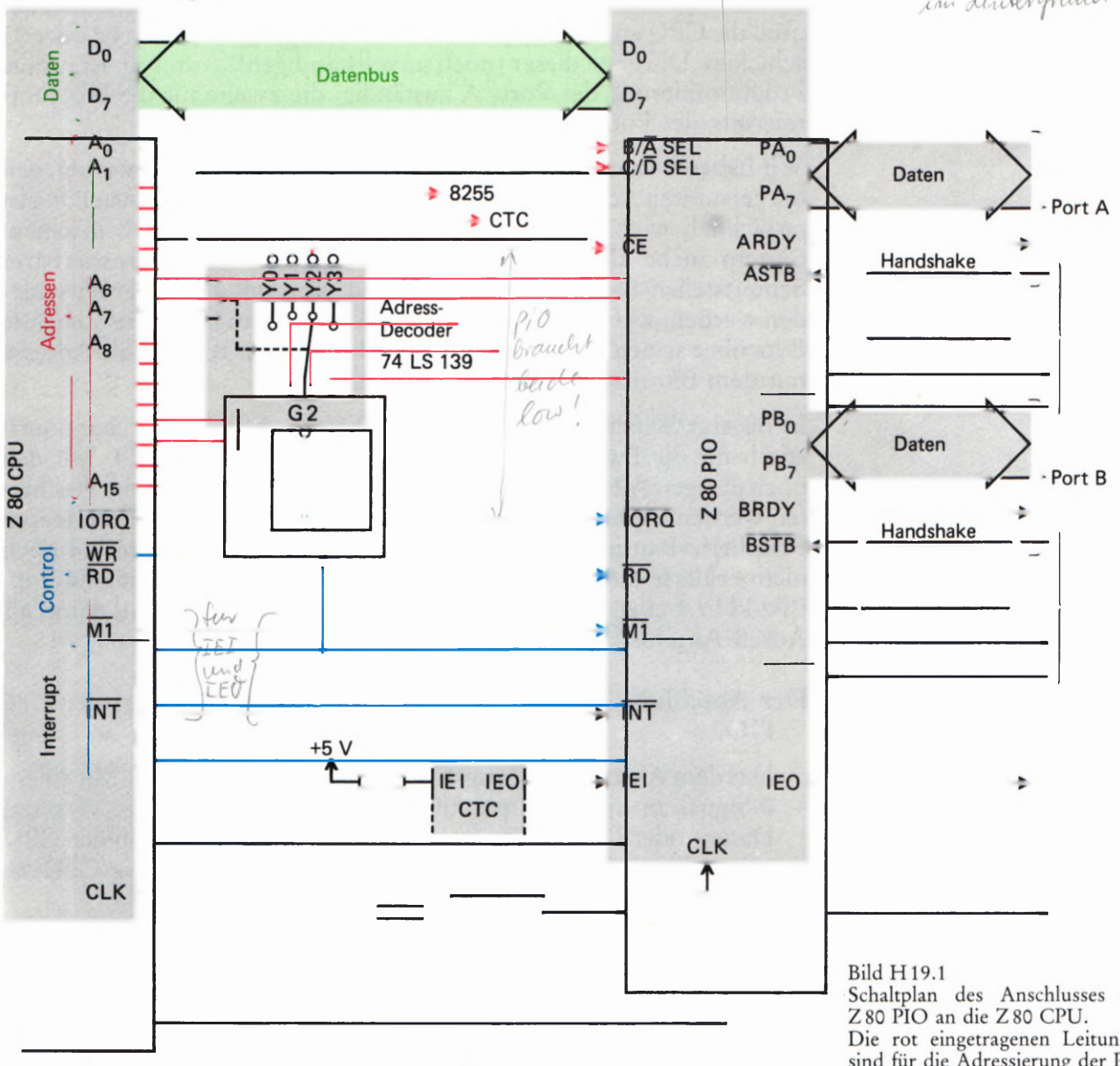


Bild H 19.1
Schaltplan des Anschlusses der Z 80 PIO an die Z 80 CPU.
Die rot eingetragenen Leitungen sind für die Adressierung der PIO zuständig.

wenn die Z 80 PIO so an die Z 80 CPU angeschlossen ist, wie es das Bild H 19.1 zeigt, und wenn der PIO-Anschluß CE zunächst unberücksichtigt bleibt?

Die Lösung dieser Aufgabe finden Sie auf der Seite U 1.

Mit den hier angestellten Überlegungen ist die erste der auf der Seite H 16 gestellten Fragen nun ebenfalls beantwortet: Die CPU kann der Z 80 PIO über den Adreßbus mitteilen, was mit den auf dem Datenbus übermittelten Bytes zu geschehen hat.

Weil es sich bei den entsprechenden Informationen offenbar um Adressen handelt (denn was anders als Adressen kann auf dem Adreßbus stehen?), wird der Sachverhalt besser anders formuliert: Die Z 80 PIO kann über vier unterschiedliche Adressen angesprochen werden. Jeder dieser vier (noch unvollständigen!) Adressen ist eine bestimmte Funktion in der Z 80 PIO zugeordnet.

Bytes, welche die PIO sozusagen unbesehen an die Peripherie weitervermitteln soll, müssen ihr an die (noch unvollständigen!) Adressen ØØ und Ø1 geschickt werden. Dabei ist die Adresse ØØ für die Bytes zuständig, die über den PIO-Port A ausgesendet werden sollen. Bytes, die über den Port B ausgesendet werden sollen, müssen (noch unvollständig!) mit Ø1 adressiert werden.

Bytes, welche für die Programmierung der Z 80 PIO vorgesehen sind, muß die CPU an eine der (noch unvollständigen!) Adressen Ø2 oder Ø3 schicken. Die erste dieser (noch unvollständigen!) Adressen ist für die Programmierung des Ports A zuständig, die zweite für die Programmierung des Ports B.

Wir haben hier sehr beharrlich immer wieder darauf hingewiesen, daß die genannten Adressen noch unvollständig sind. Diese Feststellung ist tatsächlich wichtig. Es ist ja ohne weiteres denkbar, daß in einem System mehr als nur ein auf entsprechende Weise adressierbarer Schnittstellen-Baustein verwendet wird. Wie soll dann aber entschieden werden, welcher dieser Bausteine das auf dem Datenbus stehende Byte über seinen Port B aussenden soll, wenn das Byte an die Adresse mit dem Bitmuster ØØØØ ØØØ1 geschickt wird?

In unserer Schaltung sind die Adreß-Bits Nr. Ø und Nr. 1 charakteristisch für die Funktions-Adressen innerhalb der Z 80 PIO. Mit den noch übrigen Adreß-Bits Nr. 2 bis Nr. 7 muß eine Möglichkeit geschaffen werden, einen und nur einen bestimmten von vielleicht mehreren Peripherie-Bausteinen anzusprechen. – Hier kommt der bisher noch nicht erklärte Anschluß CE der Z 80 PIO ins Spiel, der mit einer im Bild H 19.1 ebenfalls rot markierten Leitung beschaltet und damit als Adreß-Anschluß ausgewiesen ist.

Der Anschluß $\overline{CE}$ (*Chip Enable* – Baustein-Aktivierung) der Z 80 PIO.

Das dem Anschluß $\overline{CE}$ zugeführte Signal ist *active LOW:* Mit einem Ø-Signal an diesem Anschluß wird die PIO in die Lage versetzt, Daten- oder Befehls-Bytes nach einem OUT-Befehl von der CPU anzunehmen bzw. Daten nach einem IN-Befehl an die CPU zu senden.

Ein 1-Signal am Anschluß $\overline{CE}$ sperrt den Datenverkehr zwischen der PIO und der CPU.

Über den Anschluß $\overline{CE}$ wird der Z 80 PIO also die Information übermittelt, ob an ihren Anschlüssen B/$\overline{A}$ und C/$\overline{D}$ liegende Signale ausgewertet werden müssen, oder ob diese Signale irgendeine andere Einheit des Systems betreffen.

Das Bild H 19.1 zeigt, wie das funktioniert. Die Adreß-Bits A6 und A7 werden an den Eingang eines Adreß-Decoders (74 LS 139) geschaltet. Es sind insgesamt vier unterschiedliche Kombinationen der Signale A6 und A7 möglich. Jeder dieser Kombinationen entspricht eine bestimmte Stellung des (im IC natürlich elektronisch realisierten) Schalters. Je nach Stellung des Schalters wird das vom CPU-Anschluß $\overline{IORQ}$ kommende Signal auf einen der Decoder-Anschlüsse YØ bis Y3 geschaltet. Die vom Schalter gerade nicht angesprochenen Y-Anschlüsse führen 1-Signale.

Im Bild H 21.1 haben wir diese Funktion des Decoders in Form einer Tabelle dargestellt. Wenn die CPU an ihrem Anschluß $\overline{IORQ}$ ein 1-Signal liefert, dann liegen an allen Y-Anschlüssen – unabhängig von der Schalterstellung – 1-Signale.

Die folgenden vier Zeilen der Tabelle zeigen, an jeweils welchem Y-Anschluß ein Ø-Signal erscheint, wenn eine bestimmte Kombination der Adreß-Bits anliegt.

Für die Z 80 PIO interessiert das Ø-Signal am Anschluß Y2, denn nur dann, wenn dieses Signal den Wert Ø hat, wird die PIO über ihren Anschluß $\overline{CE}$ aktiviert. Und das ist nur bei der Kombination der Signale A7 = 1, A6 = Ø der Fall. – Wir haben im Bild H 19.1 angedeutet, daß bei der Adreß-Bit-Kombination A7 = Ø, A6 = Ø mit YØ = Ø in entsprechender Weise der später vorzustellende Peripherie-Baustein 8255 aktiviert wird. Bei der Adreß-Bit-Kombination A7 = Ø und A6 = 1 wird mit dem Signal Y1 = Ø der Timer-Baustein CTC aktiviert. (Vgl. Seite H 8.)

G2	A_7	A_6	YØ	Y1	Y2	Y3
1	X	X	1	1	1	1
Ø	Ø	Ø	Ø	1	1	1
Ø	Ø	1	1	Ø	1	1
Ø	1	Ø	1	1	Ø	1
Ø	1	1	1	1	1	Ø

Bild H 21.1
Funktionstabelle des Decoders 74 LS 139. – Die farbig unterlegten Werte gelten für Adressierung der Z 80 PIO.

Wir können jetzt die auf der Seite H 18 dargestellte Tabelle mit den noch unvollständigen Port-Adressen vervollständigen:

	A7	A6					C/$\overline{D}$	B/$\overline{A}$	Port-	
Bit:	7	6	5	4	3	2	1	Ø	Adr.	Information
	1	Ø	X	X	X	X	Ø	Ø	8 Ø	Daten für Port A
	1	Ø	X	X	X	X	Ø	1	8 1	Daten für Port B
	1	Ø	X	X	X	X	1	Ø	8 2	Befehl für Port A
	1	Ø	X	X	X	X	1	1	8 3	Befehl für Port B

So, wie wir den Sachverhalt bis jetzt vorgestellt haben, ergibt sich aber noch eine Schwierigkeit. – Überlegen Sie bitte einmal, was passiert, wenn die CPU den Befehl LD (1882H),A ausführt! Der Befehl verlangt, daß das im Akkumulator abgelegte Byte in der Speicherzelle mit der Adresse 1882H abgelegt wird. (Vgl. Lehrgang Mikroprozessortechnik, Seite S 43. Wir haben hier die Adresse mit dem nachgestellten Buchstaben H als sedezimal – Hexadezimal – dargestellt gekennzeichnet.)

Der Befehl LD (1882H),A setzt das Bitmuster ØØØ1 1ØØØ **1ØØØ ØØ1Ø** auf den Adreßbus und liefert dann das Byte aus dem Akkumulator über den Datenbus an die angesprochene Adresse. Die Adreß-Bits A7 bis AØ haben jetzt genau die Werte, mit denen entsprechend der oben dargestellten Tabelle die Z 80 PIO zur Entgegennahme eines Befehls-Bytes für den Port A aktiviert wird. Entsprechend der Absicht des Programmierers würde mit dem Befehl zwar das Byte aus dem Akkumulator in der adressierten Speicherzelle abgelegt, aber ganz gegen seinen Willen würde mit diesem Byte auch die Z 80 PIO zu wahrscheinlich sehr unsinnigem Verhalten programmiert.

Dieses unsinnige Verhalten verhindert das vom CPU-Anschluß $\overline{IORQ}$ kommende Signal. IORQ ist die Abkürzung für *IN/OUT ReQuest*, also für die Aufforderung, IN- und OUT-Befehle zu erkennen. Der Querstrich über der Anschlußbezeichnung im Bild H 19.1 kennzeichnet das Signal als *active LOW:* Die CPU liefert an diesen Anschluß immer dann ein Ø-Signal, wenn ein IN- oder ein OUT-Befehl ausgeführt wird. Bei der Ausführung aller anderen Befehle liefert dieser Anschluß ein 1-Signal.

Im Bild H 19.1 erkennen Sie, daß der Adreß-Decoder nur dann an einen seiner Y-Anschlüsse ein Ø-Signal liefern kann, wenn das an den Decoder-Anschluß G2 gelieferte Signal $\overline{IORQ}$ den Wert Ø hat. Zusätzlich wird das IORQ-Signal auch noch an den Anschluß $\overline{IORQ}$ der Z 80 PIO geschaltet.

Der Anschluß $\overline{\text{IORQ}}$ aktiviert mit dem angelegten Signal die Z 80 PIO nur dann, wenn dieses Signal und gleichzeitig das am Anschluß $\overline{\text{CE}}$ liegende Signal den Wert Ø hat.

Jetzt wird klar, daß der Befehl LD (18Ø2H),A die Z 80 PIO nicht berühren kann, denn es handelt sich weder um einen IN- noch um einen OUT-Befehl. Das CPU-Signal $\overline{\text{IORQ}}$ hat den Wert 1, und damit bleibt – trotz passender Adresse – die Z 80 PIO inaktiv.

Bei einem IN- oder OUT-Befehl sieht die Sache anders aus. Das Signal $\overline{\text{IORQ}}$ hat den Wert Ø. Am PIO-Anschluß $\overline{\text{IORQ}}$ erscheint ein Ø-Signal. Wenn dann noch mit der richtigen Adresse (A7 = 1 und A6 = Ø) an den Anschluß $\overline{\text{CE}}$ gleichzeitig ein Ø-Signal geliefert wird, dann wird die PIO aktiviert. Stimmt dagegen die Adresse nicht ($\overline{\text{CE}}$ = 1), dann nützt auch das Ø-Signal am PIO-Anschluß nichts: Die PIO bleibt inaktiv.

Sehen Sie sich bitte noch einmal die Adressen-Tabelle auf der Seite H 21 und das Bild H 19.1 an! Beim Adressieren der PIO sind nur die Werte der Bits Nr. Ø und Nr. 1 sowie die der Bits Nr. 6 und Nr. 7 von Interesse. Die Bits Nr. 2 bis Nr. 5 können beliebige Werte annehmen *(don't care)*. Das bedeutet, daß z. B. Daten für den PIO-Port B nicht nur mit 81 adressiert werden können. Diese Daten werden von der PIO auch dann richtig empfangen, wenn sie von der CPU z. B. an die Adresse B9 mit dem Bitmuster **1Ø**11 1**ØØ**1 oder an die Adresse A6 mit dem Bitmuster **1Ø**1Ø Ø11**Ø** geschickt werden. Wichtig sind nur die Werte der hier jeweils stärker gedruckten Bits.

Man sagt: Die Port-Adressen der PIO sind nur unvollständig decodiert (vgl. Seite H 18). Mit dieser unvollständigen Decodierung spart man einigen Aufwand in der Hardware. Nachteilig ist allerdings, daß damit eine Reihe anderer Port-Adressen blockiert werden. Sie können im System nicht mehr zum Adressieren anderer Peripherie verwendet werden.

Aufgabe H 22.1

Überlegen Sie bitte, wieviele Port-Adressen durch die unvollständige Adressen-Decodierung der Z 80 PIO für andere Verwendungen unbrauchbar werden. Welche Adressen sind das?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 2.

Im Micro-Professor-System bedeutet die unvollständige Decodierung der Port-Adressen keine Einschränkung, weil nur eine beschränkte Anzahl von Peripherie-Bausteinen adressiert werden muß.

Die Centronics-Schnittstelle

Drucker für den Anschluß an Mikroprozessor-Systeme werden meist wahlweise mit serieller oder mit paralleler Schnittstelle angeboten. Was es damit auf sich hat, haben wir ab der Seite H 3 erläutert: Bei einer seriellen Schnittstelle müssen die Bytes für die druckbaren Zeichen vom System Bit-seriell, Byte-seriell an den Drucker übermittelt werden; bei einer parallelen Schnittstelle werden die Zeichen Bit-parallel, Byte-seriell übermittelt.

Wenn man bereits ein System besitzt, an das ein Drucker angeschlossen werden soll, dann muß man sich vor dem Kauf des Druckers informieren, welche Art von Schnittstelle das System zur Verfügung stellt, damit auch der Drucker mit der entsprechenden Schnittstelle ausgerüstet ist.

Und was ist zu tun, wenn das eigene System weder eine serielle noch eine parallele Schnittstelle hat, die für den Anschluß eines Druckers vorbereitet ist? – In diesem Fall muß man sich die Schnittstelle eben selbst einrichten. Wir wollen Ihnen in diesem Abschnitt zeigen, daß das durchaus kein Hexenwerk ist.

Das Prinzip einer seriellen Schnittstelle haben wir Ihnen bereits am Anfang des ersten Lehrbriefs vorgestellt. Im dritten Lehrbrief werden Sie sehen, wie man durch die kombinierte Benutzung der Z 80 PIO und des Z 80 CTC sehr elegant eine serielle Schnittstelle einrichten kann, mit der bei Bedarf auch an Ihrem Micro-Professor ein Drucker mit ebenfalls serieller Schnittstelle arbeitet.

Hier wollen wir uns mit einer parallelen Schnittstelle beschäftigen. – In den Angeboten für Drucker finden Sie häufig statt der Angabe „parallele Schnittstelle" die Bezeichnung „Centronics-Schnittstelle". Was hat es damit auf sich?

Bei der Erläuterung der seriellen Schnittstelle haben wir Ihnen gezeigt, daß ein System immer erst dann ein Byte an den Drucker senden darf, wenn der Drucker mit einem CTS-Signal seine Bereitschaft bekundet, dieses Signal auch verarbeiten zu können (Seite H 4). Inzwischen wissen Sie, daß dieses CTS-Signal nichts anderes als ein Handshake zwischen dem System und dem Drucker bedeutet (vgl. Seite S 3): Bei der Daten-Kommunikation muß der Drucker als Empfänger dem sendenden System jeweils mitteilen, ob er nach dem Empfang des vorhergehenden Zeichens in der Lage ist, das nächstfolgende Zeichen aufzunehmen und zu verarbeiten.

Das entsprechende Prinzip gilt auch bei der (Bit-)parallelen, Byte-seriellen Datenübertragung: Das jeweils nächstfolgende Byte darf das System erst dann an den Drucker übertragen, wenn der Drucker das vorhergehende Byte verarbeitet hat und auch sonst keine Bedenken bestehen, daß im Drucker ein Byte verloren gehen könnte. Solche Bedenken gibt es z. B. dann, wenn der Drucker eine Seite vollgeschrieben hat und kein neues Papier eingelegt worden ist.

Wir haben Ihnen gezeigt, daß man bei der seriellen Datenübertragung mit zwei Signalleitungen auskommt: Mit der Datenleitung vom System zum Drucker und mit der Quittungsleitung (CTS) vom Drucker zum System. Bei der parallelen Datenübertragung braucht

man immer sehr viel mehr Leitungen. Da sind einmal die acht Datenleitungen für die Daten-Bits und zumindest eine zusätzliche Leitung für ein Quittungs-Signal. Wenn das nun schon so ist, dann kommt es auf eine oder zwei zusätzliche Leitungen auch nicht mehr an. Der Handshake-Betrieb kann also bei parallelen Schnittstellen fast beliebig komfortabel gestaltet werden.

Einen Kompromiß hat in dieser Hinsicht erstmalig die amerikanische Firma *Centronics* für ihre Drucker geschlossen. Diesem Kompromiß haben sich inzwischen fast alle Drucker-Hersteller angeschlossen, so daß eine solche Schnittstelle fast zum Standard geworden ist. Exakt muß man eine solche Schnittstelle demnach als **Centronics kompatibel** (passend zu Druckern der Firma Centronics) bezeichnen.

Das Handshake bei der Centronics-Schnittstelle

Im Bild H25.1 haben wir skizziert, wie ein Drucker mit Centronics-Schnittstelle mit den Anschlüssen der Ports A und B der Z 80 PIO verbunden werden kann. Dieses Anschluß-Schema werden wir nachher noch näher erläutern. Hier interessieren zunächst nur die rechts im Bild dargestellten Anschlüsse des Druckers.

Sie erkennen zunächst die Datenbit-Anschlüsse DB0 bis DB7, über welche die acht Bits eines Bytes vom System parallel an den Drucker übermittelt werden. Außerdem kommt vom Port B des Systems ein Signal, das dem Drucker mitteilt, daß er jetzt die an die Datenbit-Anschlüsse gelieferten Bits einlesen kann. Der Drucker faßt dieses Signal als STROBE-Signal auf. – Die Signal-Leitungen, über die das System Daten zum Drucker schickt, haben wir in unserem Bild grün eingetragen.

Die im Bild H25.1 unten dargestellten Leitungen führen Signale, die der Drucker an das System schickt. Bei diesen Signalen handelt es sich um Quittungs-Signale, über die das Handshake abgewickelt wird. Zwei dieser Leitungen haben wir zur Kennzeichnung der Signal-Richtung rot eingetragen. Die dritte Leitung (Paper Out) müßte eigentlich ebenfalls rot eingetragen werden. Wir haben das nicht getan, weil

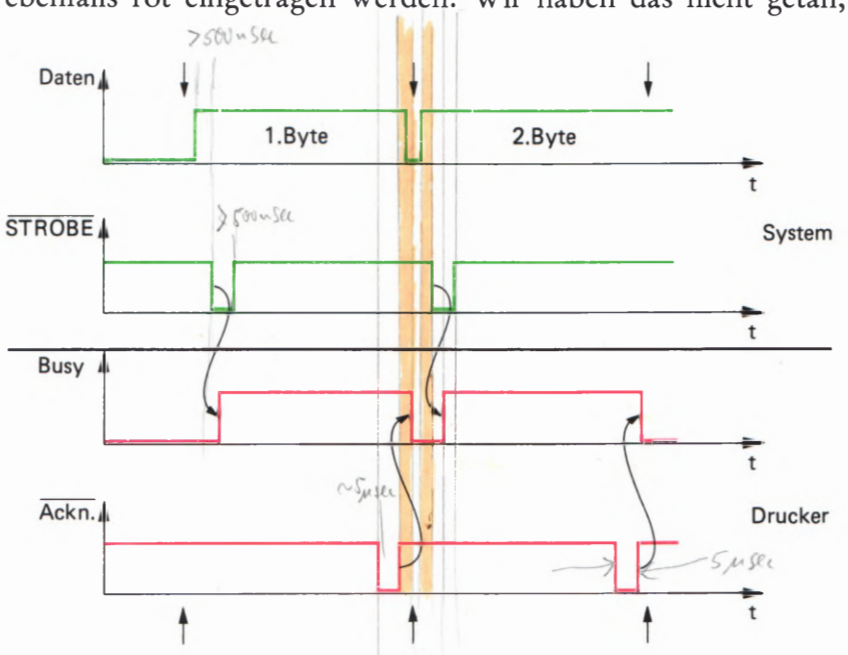


Bild H24.1
Zeitlicher Zusammenhang der zwischen Drucker und Mikroprozessor-System über eine Centronics-Schnittstelle ausgetauschten Signale.

dieses Signal zunächst von untergeordneter Bedeutung ist.

Die Centronics-Schnittstelle eines Druckers verfügt manchmal noch über einige weitere Anschlüsse. Wir betrachten hier nur die Anschlüsse, die mit Sicherheit immer vorhanden sind.

Sehen Sie sich jetzt bitte das Bild H24.1 an! Dort haben wir den zeitlichen Zusammenhang der wichtigsten Signale eingetragen, die bei einer Centronics-Schnittstelle verwendet werden. Die Farben entsprechen denen im Bild H25.1: Grün sind Signale vom System an den Drucker eingetragen, rot solche Signale, die der Drucker an das System schickt.

Zunächst wollen wir die Bezeichnungen der einzelnen Signale erläutern. Daß mit **Daten** die Gesamtheit der acht Bits eines Bytes gemeint ist, bedarf keiner Erklärung.
Auch die Bedeutung des **STROBE-Signals** haben wir bereits gezeigt. Der Drucker erwartet nach der Bereitstellung der acht Datenbits vom System dieses *active LOW*-Signal als Mitteilung, daß die acht Datenbits jetzt gültig sind. Das heißt: Frühestens 0,5 µs nach der Bereitstellung der acht Datenbits muß das System über die entsprechende Leitung ein mindestens 0,5 µs langes Ø-Signal an den Drucker schikken. Erst dann übernimmt der Drucker die auf den Datenleitungen stehenden acht Bits eines Bytes.

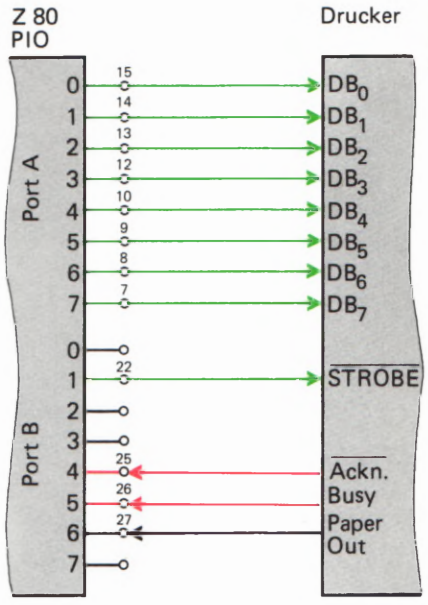


Bild H25.1
Anschluß eines Druckers mit Centronics-Schnittstelle an den PIO CTC I/O BUS des Micro-Professors. Die eingetragenen Ziffern beziehen sich auf die Bezeichnungen im Bild H28.1.

Hier ist ein Hinweis angebracht. In Bild S3.1 finden Sie ebenfalls eine Leitung, die ein STROBE-Signal führt, aber dieses Signal wird nicht vom System an die Peripherie geschickt, sondern von der Peripherie an das System. Das ist kein Widerspruch zu der soeben beschriebenen Feststellung. STROBE ist ganz allgemein die Bezeichnung für ein Signal, das eine immer wiederkehrende, kurzzeitige Information für irgendeinen Zustand enthält. Es ist zwar nicht geschickt, innerhalb ein und derselben Anwendung zwei unterschiedliche Signale auf die gleiche Weise zu bezeichnen (auch wenn sie im Prinzip entsprechende Bedeutungen haben). Wir konnten uns jedoch nicht entschließen, unterschiedliche Signalbezeichnungen zu wählen, weil Sie in den Datenblättern doch wieder die Bezeichnungen STROBE finden.

Die in den Bildern gewählte Abkürzung **Ackn.** steht für das englische Wort *Acknowledge* (sprich: äcknollitsch). Wörtlich ist damit eine Empfangsbestätigung gemeint. Das so bezeichnete Signal ist *active LOW* (quergestrichen!). Es wird normalerweise vom Drucker mit dem Wert 1 an das System geliefert. Wenn der Drucker ein Byte vom System eingelesen und verarbeitet hat, dann meldet er diese Tatsache an das System mit einem kurzzeitigen (etwa 5 µs langen) Ø-Signal.

Das englische Wort **Busy** (sprich: bisi) bedeutet: Beschäftigt. Das so bezeichnete Signal ist *active HIGH*. – Immer dann, wenn der Drucker nicht gerade busy ist, wenn er also nicht damit beschäftigt ist, ein vom System übermitteltes Byte zu verarbeiten, sendet er ein Signal Busy = Ø an das System. Sobald er aber ein Byte vom System eingelesen hat, setzt er das Signal Busy auf den Wert 1, und solange das der Fall ist, darf das System kein neues Byte an den Drucker schicken.

Jetzt können wir daran gehen, uns den zeitlichen Zusammenhang der in Bild H24.1 dargestellten Signale anzusehen.

Das als oberes dargestellte Diagramm (Daten) stellt keinen Signal-Verlauf dar. Es wird dort lediglich gezeigt, zu welcher Zeit die acht Datenbits vom System an die DB-Anschlüsse des Druckers geliefert werden, und wann die Werte dieser Datenbits wechseln.

Zu irgendeinem Zeitpunkt liefert das System die Bits des ersten Bytes an den Drucker. Einen Augenblick später sendet das System ein STROBE-Ø-Signal aus und macht damit die Datenbits an den DB-Anschlüssen des Druckers gültig.

Der geschweifte Pfeil zwischen dem Signalverlauf des STROBE-Signals und dem Signalverlauf des Busy-Signals macht deutlich, wie der Drucker jetzt reagiert: Der 1-Ø-Übergang am Anfang des STROBE-Signals löst unmittelbar den Ø-1-Übergang des Busy-Signals aus, das der Drucker an das System schickt. Das System muß dem Busy-1-Signal die Information entnehmen, daß es zunächst kein weiteres Daten-Byte an den Drucker schicken darf.

Nach einiger Zeit hat der Drucker dann das soeben eingelesene Byte verarbeitet: Entweder hat er das entsprechende Zeichen zu Papier gebracht, oder er hat – wenn er dafür eingerichtet ist – das übermittelte Zeichen in einem internen Speicher zwischenzeitlich abgelegt, so daß er dem System das nächstfolgende Byte bereits abnehmen kann, auch wenn er das vorhergehende Zeichen noch nicht zu Papier gebracht hat.

Für das System ist diese interne Arbeitsweise des Druckers nicht weiter wichtig. Hier ist nur interessant, ob der Drucker das nächstfolgende Zeichen abnehmen kann oder nicht.

Sobald der Drucker mit seiner internen Verarbeitung eines eingelesenen Zeichens fertig ist, sendet er auf einer besonderen Leitung das kurzzeitige Acknowledge-Ø-Signal aus. Erst mit dem Ø-1-Übergang am Ende dieses Acknowledge-Signals wird das Busy-Signal gelöscht, und jetzt darf das System das nächstfolgende Byte an den Drucker schicken.

In Bild H24.1 haben wir durch senkrechte Pfeile oberhalb und unterhalb der Diagramme die Zeitpunkte markiert, bei denen das System das jeweils nächstfolgende Byte über einen Port an den Drucker schikken und anschließend mit einem STROBE-Signal gültig machen darf. Sehen Sie sich bitte an, wann das der Fall ist. (Nehmen Sie ein Lineal zu Hilfe!)

Am einfachsten ist es, wenn das System darauf wartet, daß der Drucker ein Acknowledge-Ø-Signal aussendet, denn mit diesem Signal meldet der Drucker, daß er das vorher übermittelte Byte verarbeitet hat. Wir werden Ihnen noch eine Methode vorstellen, bei der das System tatsächlich nur dieses Acknowledge-Signal auswertet.

In der Praxis hat das System allerdings bei der Abfrage des Acknowledge-Signals Schwierigkeiten, weil dieses Signal ja nur etwa 5 µs lang ist (vgl. Seite H5). Das System kann die Abfrage nur in einer Programm-Schleife mit einer immer wiederholten IN-Anweisung vornehmen. Dabei besteht die Gefahr, daß das sehr kurze Acknowledge-Signal einfach übersehen wird.

Wesentlich sicherer ist es, das vom Drucker ausgesendete Busy-Signal abzufragen: Solange das Busy-Signal den Wert 1 hat, darf kein neues Byte übermittelt werden. Erst dann, wenn das Busy-Signal den Wert Ø annimmt, ist sichergestellt, daß der Drucker ein neues Byte verarbeiten kann.

In der Software für eine Centronics-Schnittstelle, die wir Ihnen als erste vorstellen, lassen wir das System sowohl das Acknowledge- als auch das Busy-Signal abfragen: Ein neues Byte darf das System immer dann an den Drucker übermitteln, wenn das Acknowledge-Signal den Wert 1 und gleichzeitig das Busy-Signal den Wert Ø hat.

Wenn wir dann schon zwei Signale, die vom Drucker kommen, vom System abfragen lassen, dann macht es auch nichts mehr aus, auch noch ein drittes Signal des Druckers auszuwerten. Es handelt sich um das Paper-Out-Signal. Dieses Signal des Druckers hat normalerweise den Wert Ø. Nur dann, wenn der Drucker unter dem Schreibkopf kein Papier mehr vorfindet, wird das Paper-Out-Signal auf den Wert 1 gesetzt.

Für die Übermittlung eines Bytes vom System an den Drucker gelten jetzt folgende Bedingungen für die vom Drucker gelieferten Handshake-Signale:

Acknowledge = 1
Busy = Ø
Paper Out = Ø

Anschluß eines Druckers mit Centronics-Schnittstelle

Das Bild H25.1 zeigt, daß für den Anschluß eines Druckers mit Centronics-Schnittstelle entsprechend den soeben angestellten Überlegungen ein einzelner 8-Bit-Port nicht ausreicht. (Wir werden Ihnen noch zeigen, wie man beim Ausnutzen aller Möglichkeiten der Z 80 PIO letztlich doch mit einem einzigen Port auskommen kann.)

Die acht Daten-Anschlüsse eines Ports werden für die parallele Übertragung der acht Bits eines Bytes benötigt. Mit den Handshake-Leitungen (STROBE, Acknowledge, Busy und Paper Out) werden zusätzlich vier weitere Anschlüsse des zweiten Ports der PIO belegt. Welche der acht Port-Anschlüsse für diese Handshake-Leitungen verwendet werden, ist grundsätzlich gleichgültig. Wir haben in unserer Darstellung ganz willkürlich für das STROBE-Signal das Bit Nr. 1 verwendet und für die vom Drucker gelieferten Signale die Bits Nr. 4, 5 und 6.

Entscheidend wichtig ist die Richtung der Signale bei der Centronics-Schnittstelle. Bei den acht Datenbits des zu übertragenden Bytes muß der Port für alle Bits als Sender arbeiten. Die Handshake-Signale, die über den anderen Port ausgetauscht werden, haben unterschiedliche Richtungen. Das STROBE-Signal wird vom System an den Drucker geschickt; der für dieses Signal gewählte Anschluß muß also als Sender arbeiten. Die Signale Acknowledge, Busy und Paper Out schickt der Drucker an das System, so daß die für diese Signale gewählten Port-Anschlüsse als Empfänger arbeiten müssen.

Hier kommen uns die vielfältigen Programmier-Möglichkeiten der Z 80 PIO entgegen: Jeder Port-Anschluß der PIO kann in der Betriebsart 3 beliebig als Sender oder als Empfänger programmiert werden. Eben nur weil das möglich ist, hat man bei der Belegung der Port-Anschlüsse für die Handshake-Signale jede Freiheit.

Ehe wir uns mit der Programmierung der PIO für die Centronics-Schnittstelle befassen, wollen wir Ihnen noch zeigen, wie bei Bedarf der Drucker tatsächlich an die Z 80 PIO im Micro-Professor angeschlossen werden kann.

Aus Platzgründen stehen an der Klemmleiste der zu Ihrem Lehrgang gehörenden Peripherie-Leiterplatte nur die Anschlüsse eines der beiden Ports der Z 80 PIO zur Verfügung (Port B, vgl. Bild H 12.1). Der Drucker muß also direkt an die mit PIO CTC I/O BUS bezeichnete, 40-polige Stiftleiste auf dem Micro-Professor angeschlossen werden. Im Bild H 28.1 haben wir die Belegung dieser Stiftleiste dargestellt. Die in diesem Zusammenhang interessierenden Port-Anschlüsse haben wir außen angeschrieben. Was es mit den – abgesehen von den GND- (= Ground, Masse-)Anschlüssen – übrigen Anschlüssen auf sich hat, erläutern wir im Zusammenhang mit dem Z 80 CTC.

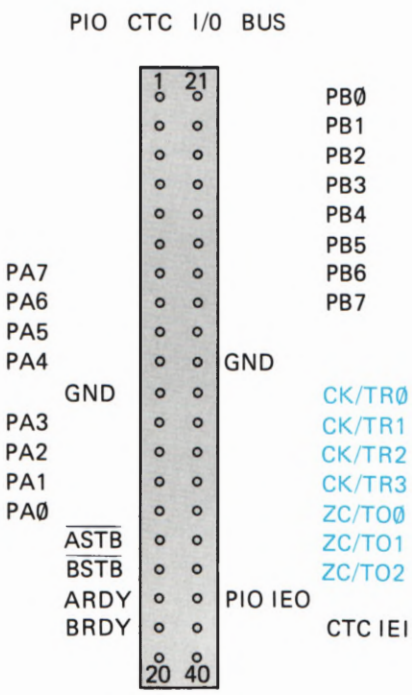


Bild H 28.1
Anschlußbelegung der Steckerleiste für den PIO CTC I/O BUS des Micro-Professors. Die seitlich herausgestellten Bezeichnungen werden für den Anschluß eines Druckers benötigt.

In Bild H 25.1 finden Sie bei der Z 80 PIO die Anschluß-Nummern der PIO CTC I/O BUS-Stiftleiste eingetragen. Die Anschlüsse am Drukker haben wir nicht eingetragen, da diese Anschlüsse bei unterschiedlichen Druckern verschieden angeordnet sein können. Sie sind aber immer aus dem Datenblatt des betreffenden Druckers ersichtlich. Nicht vergessen werden darf natürlich die Verbindung der GND-Leitung mit dem Anschluß für das Bezugspotential beim Drucker.

Wir wollen noch einmal darauf aufmerksam machen, daß das Anschlußschema für den Drucker in unserem Beispiel mehr oder weniger willkürlich gewählt worden ist. Selbstverständlich kann man auch den PIO-Port B für die Übertragung der acht Bits eines Bytes verwenden und den Port A für die Handshake-Signale. Ebenso kann die Zuordnung der Handshake-Signale auch zu anderen Anschlüssen des entsprechenden Ports erfolgen. Diese Zuordnungen müssen dann natürlich bei der nachher vorgestellten Programmierung der PIO und beim Programm für die Daten-Ausgabe berücksichtigt werden.

Initialisierung der PIO für eine Centronics-Schnittstelle

Wie bereits bei den Programmen zu den Versuchen S 12.1 und 15.1 wollen wir die Programmierung der Z 80 PIO in einem speziellen Unterprogramm vornehmen, das als INIT bezeichnet wird.

Die Programmierung erweist sich als sehr einfach. Beide Ports sollen in der Betriebsart 3 arbeiten. Das ist zwar beim Port A, über den wir die acht Datenbits eines Bytes ausgeben wollen, nicht unbedingt notwendig: Dieser Port könnte auch in der Betriebsart Ø als Sender für alle acht Datenbits arbeiten. In diesem Fall könnte man sich sogar das I/O-Register-Steuerwort sparen, und ein Interrupt-Vektor-Wort ist erst recht nicht notwendig. (Vgl. die Seiten S 3 und S 8).

Der Einheitlichkeit halber programmieren wir beide Ports für die gleiche Betriebsart. Wenn das geschieht, dann müssen alle acht Datenanschlüsse des Ports A mit dem I/O-Register-Steuerwort ØØ als Sender (Ausgänge) programmiert werden (Bild S 9.1).

Der Port B muß zur Verarbeitung der Handshake-Signale unbedingt zum Arbeiten in der Betriebsart 3 programmiert werden, denn nur dann lassen sich die Port-Anschlüsse einzeln beliebig als Sender oder als Empfänger einstellen.

Für die von uns (willkürlich) gewählte Belegung der Anschlüsse des Ports B mit den Handshake-Signalen (Bild H 25.1) ergibt sich das Bitmuster des I/O-Register-Steuerworts entsprechend der Darstellung im Bild H 29.1. Die Bits Nr. Ø, 2, 3 und 7 haben *Don't Care*-Werte. Das Bit Nr. 1 mit dem Wert Ø macht den entsprechenden Port-Anschluß zum Ausgang (Sender) für das STROBE-Signal. Mit den 1-Bits Nr. 4, 5 und 6 werden die Anschlüsse für die Signale Acknowledge, Busy und Paper Out als Eingänge (Empfänger) programmiert.

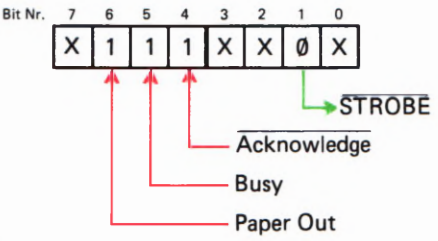


Bild H 29.1
Bitmuster des I/O-Register-Steuerworts für den Anschluß eines Druckers entsprechend dem Bild H 25.1.

Für die Programmierung der PIO ergibt sich demnach folgender Programm-Baustein:

```
INIT: LD   A,ØCFH          ;Port A: Betriebsart 3
      OUT (PIOBA),A

      LD   A,ØØØØØØØØB     ;        Alle Anschlüsse Ausgänge
      OUT (PIOBA),A

      LD   A,ØCFH          ;Port B: Betriebsart 3
      OUT (PIOBB),A


      LD   A,11111101B     ;        Bit Nr. 1 STROBE  Ausgang
                           ;        Bit Nr. 4 Ackn.   Eingang
                           ;        Bit Nr. 5 Busy    Eingang
                           ;        Bit Nr. 6 Paper Out Eingang
      OUT (PIOBB),A

      RET                  ;Ende der Routine
```

Bei dieser mnemonischen Befehls-Formulierung haben wir das gleiche Schema verwendet, das wir Ihnen bereits auf der Seite S 13 beim Flipflop-Programm vorgestellt haben. PIOBA und PIOBB sind symbolische Bezeichnungen für die CPU-Port-Adressen 82H und 83H, die im Micro-Professor-System für die PIO-Befehle zuständig sind. Im Programm müssen diese symbolischen Bezeichnungen selbstverständlich vorab mit EQU-Anweisungen definiert werden. (Vgl. Seite L 2.)

Wir stellen Ihnen nachher ein Programm vor, in dem der Micro-Professor Daten über eine Centronics-Schnittstelle ausgibt. In diesem Programm finden Sie die eben beschriebene INIT-Routine zur Programmierung der Z 80 PIO (Seite L 19).

Aufgabe H 29.1

Das im Bild H 25.1 dargestellte Anschlußschema eines Druckers mit Centronics-Schnittstelle haben wir recht willkürlich gewählt (vgl. Seite H 28).

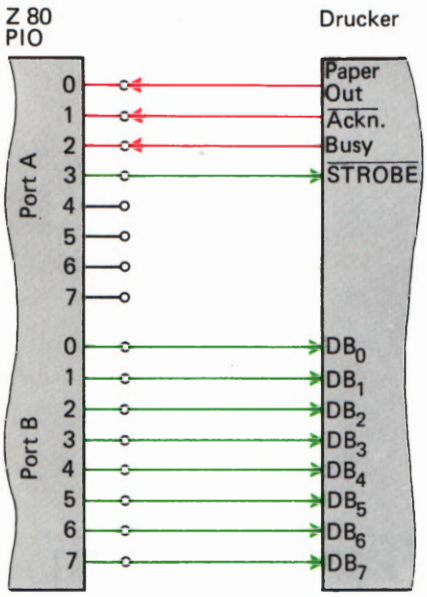


Bild H30.1
Zu Aufgabe H29.1: Die Z80PIO soll so programmiert werden, daß sie das hier dargestellte Anschlußschema eines Druckers berücksichtigt.

Schreiben Sie bitte eine INIT-Routine zur Programmierung der Z 80 PIO für den Fall an, daß der PIO-Port B für die Übertragung der acht Bits eines Bytes verwendet wird. Das Handshake soll über den PIO-Port A abgewickelt werden. Dabei werden die Signalleitungen wie folgt angeschlossen:

Paper Out	Bit Nr. Ø
Acknowledge	Bit Nr. 1
Busy	Bit Nr. 2
STROBE	Bit Nr. 3

In Bild H30.1 haben wir das entsprechende Anschlußschema dargestellt.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü3.

Die Codierung von Druckzeichen

Eine Routine zum Drucken von Zeichen ist natürlich nur dann sinnvoll, wenn auch etwas zum Drucken da ist. Wir haben Ihnen ab der Seite L17 ein kurzes Programm aufgelistet, das den Inhalt eines definierbaren Speicherbereichs als sogenannten **Hex-Dump** formatiert. Das Bild H30.2 zeigt, was damit gemeint ist: Die zwischen der Anfangsadresse und der Endadresse des definierten Speicherbereichs stehenden Bytes werden zeilenweise hintereinander ausgegeben. In jeder Zeile stehen sechzehn (in manchen Fällen auch nur vier oder acht) Bytes; am Anfang der Zeile wird jeweils die Adresse des ersten in der Zeile stehenden Bytes angegeben.

Das Programm ist im hier betrachteten Zusammenhang nicht weiter interessant. Wenn Sie mögen, dann können Sie seine Arbeitsweise anhand der Kommentare in der Programm-Auflistung studieren. Wir wollen uns aber dafür interessieren, wie und in welcher Form das Programm die aus dem Speicher geholten Bytes an den Drucker übermittelt.

Im Speicher des Micro-Professors abgelegte Bytes sind Daten, die in Form von Bitmustern von der CPU verarbeitet werden. Zur gedruckten Darstellung solcher Daten werden jeweils zwei Druckzeichen benötigt. Z. B. wird das Bitmuster 1Ø1Ø Ø111 als Byte durch die beiden Druckzeichen A7 dargestellt. Unter Druckzeichen wollen wir Ziffern und Buchstaben, sowie Satz- und Sonderzeichen verstehen. Zu den Sonderzeichen gehören z. B. ein Stern (*) oder ein nach oben weisender Pfeil (↑).

Bild H30.2
So sieht der von einem Nadeldrukker gelieferte Hex-Dump des Programms auf den Seiten L17 bis L19 aus.

```
1800  CD 7F 18 21 00 18 11 8F 18 7C CD 47 18 7D CD 47
1810  18 3E 20 CD 65 18 CD 65 18 06 10 7E CD 47 18 3E
1820  20 CD 65 18 CD 36 18 23 10 F1 3E 0D CD 65 18 3E
1830  0A CD 65 18 18 D3 7B BD C0 7A BC C0 3E 0D CD 65
1840  18 3E 0A CD 65 18 C7 C5 4F 1F 1F 1F 1F CD 5C 18
1850  CD 65 18 79 CD 5C 18 CD 65 18 C1 C9 E6 0F C6 90
1860  27 CE 40 27 C9 C5 4F DB 81 E6 70 EE 10 20 F8 79
1870  D3 80 E3 E3 3E 00 D3 81 3E 02 D3 81 79 C1 C9 3E
1880  CF D3 82 3E 00 D3 82 3E CF D3 83 3E FD D3 83 C9
```

Der Mikroprozessor kann mit „A7" nicht das geringste anfangen. Die sedezimale Schreibweise von Bytes ist lediglich ein Hilfsmittel zur gedanklich bequemeren Handhabung von Mustern aus 8 Bits.

Wenn ein Drucker die Sedezimal-Darstellung des Bytes 1Ø1Ø Ø111 als A7 oder die des Bytes 11ØØ 11Ø1 als CD drucken soll, dann ist für ihn diese Zeichenfolge nichts anderes als eben eine Zeichenfolge. Ob CD ursprünglich das Bitmuster 11ØØ 11Ø1 oder die Kennzeichnung des Autos eines Diplomaten darstellt, ist dem Drucker völlig gleichgültig.

Da andererseits die interne Steuerelektronik eines Druckers einen eigenen Mikroprozessor enthält, wählt man zum Ansteuern des Drukkers Bitmuster: Jedem Zeichen, das der Drucker drucken soll, entspricht ein spezielles Bitmuster (Byte).

In den drei rechten Spalten der Tabelle auf der Seite T 1 finden Sie die 96 Zeichen zusammengestellt, die üblicherweise von einem Drucker dargestellt werden können. Es sind die Ziffern Ø bis 9, die kleinen und großen Buchstaben A bis Z und zusätzlich Satz- und Sonderzeichen.

Die in der Tafel dargestellte Zuordnung von Druckzeichen zu Bytes wird als ASCII bezeichnet (vgl. Seite H 3). Mit ASCII sind außer den druckbaren Zeichen selbst auch noch eine Reihe von Steuerzeichen (Bytes ØØ bis 1F) codiert, die wir der Vollständigkeit in die Tabelle mit aufgenommen haben. Von diesen Steuerzeichen sind hier nur die Zeichen LF (Byte ØA) und CR (Byte ØD) interessant. Das Byte ØA (**L***ine* **F***eed*) veranlaßt den Drucker, das Papier um eine Zeile weiter zu schieben. Das Byte ØD (**C***arriage* **R***eturn*) bedeutet, daß das nächstfolgende, druckbare Zeichen am Anfang einer Zeile stehen soll.

Sehen Sie sich bitte an, worin die Tätigkeit des Hex-Dump-Programms besteht, wenn es ein Byte nach dem anderen aus dem Speicher Ihres Systems holt und diese Bytes als druckbare Zeichen an den Drucker liefert:

1865			1866			1867			1868		Adressen
11ØØ	Ø1Ø1		Ø1ØØ	1111		11ØØ	11Ø1		Ø111	1Ø1Ø	Inhalte
C	5		4	F		C	D		7	A	Sedezimal-Darst.
43	35	20	34	46	20	43	44	20	37	41	ASCII
C	5		4	F		C	D		7	A	Druckzeichen

In der oberen Zeile der Tabelle sind die Adressen der Speicherzellen angeschrieben, deren Inhalte in sedezimaler Schreibweise vom Drukker dargestellt werden sollen. Darunter stehen jeweils die acht Bits, die in der zugehörigen Speicherzelle abgelegt sind. Wiederum darunter stehen in der dritten Zeile die Sedezimal-Zeichen für jeweils vier Bits, die der Drucker als Darstellung der tatsächlichen Speicherzellen-Inhalte drucken soll.

Wenn der Drucker diese Zeichen als Ziffern und Buchstaben drucken soll, müssen entsprechend dem jeweiligen Muster von vier Bits die ASCII-Bytes generiert werden. Immer zwei ASCII-Bytes sind zur Darstellung des Inhalts einer Speicherzelle notwendig.

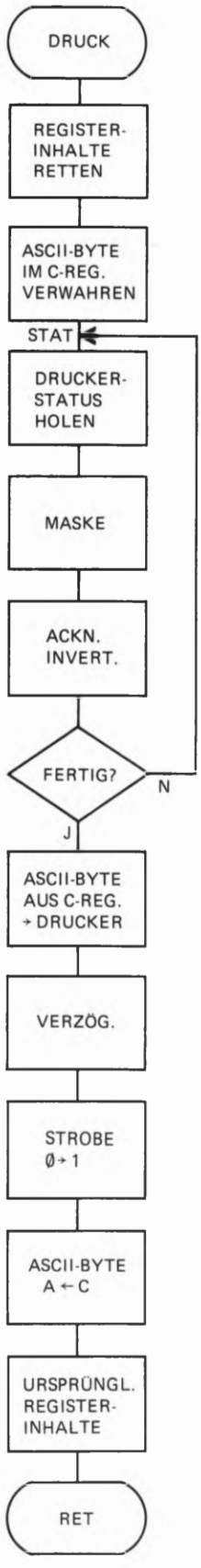


Bild H32.1
Ablaufplan der DRUCK-Routine.

Damit in der gedruckten Darstellung die Bytes lesbar voneinander getrennt sind, muß nach je zwei ASCII-Ziffern-Bytes ein ASCII-Abstands-Byte (Space, 20H) an den Drucker geliefert werden. Wir haben diese ASCII-Space-Bytes zur Verdeutlichung farbig eingetragen.

Im Hex-Dump-Programm wird die Umwandlung eines Bytes im Akkumulator in zwei ASCII-Byte von der BYTDR-Routine übernommen. Innerhalb dieser Routine wird zweimal ein ASCII-Byte der DRUCK-Routine übergeben: Das ASCII-Byte steht im Akkumulator, und anschließend wird jeweils die DRUCK-Routine aufgerufen, die das Byte aus dem Akkumulator über die PIO an den Drucker liefert.

Beachten Sie bitte, daß die ASCII-Bytes für den Abstand (Space, 20H) und für den Zeilenwechsel (**C***arriage* **R***eturn*, ØCH, und **L***ine* **F***eed*, ØAH) im Programm direkt in den Akkumulator geladen und dann der DRUCK-Routine übergeben werden.

Die Centronics-Druck-Routine

Außer dem auf den vorangegangenen Seiten beschriebenen Hardware-Anschluß des Druckers gehört zur Centronics-Schnittstelle noch eine Routine, die ein im Akkumulator stehendes ASCII-Byte über den Schnittstellen-Baustein an den Drucker liefert. Diese Routine muß jedesmal dann aufgerufen werden, wenn das System ein druckbares Zeichen abliefern will.

Die Routine muß zunächst feststelllen, ob der Drucker bereit ist, ein Zeichen zu übernehmen. Ist das nicht der Fall, dann muß die Routine das Zeichen verwahren und warten, bis der Drucker das Zeichen abnehmen kann. Sobald der Drucker seine Bereitschaft signalisiert, daß er ein Zeichen abnehmen kann, schickt die Routine das Zeichen ab, wartet einen kurzen Augenblick und macht dieses Zeichen dann durch ein STROBE-Signal gültig.

Wenn die Druck-Routine ihre Aufgabe erledigt hat, dann muß sie dem aufrufenden Hauptprogramm die Register-Inhalte wieder so hinterlassen, wie sie sie vorgefunden hat.

Im Bild H32.1 haben wir den Ablaufplan der Druck-Routine dargestellt. Er bedarf nur weniger Erläuterungen. Vergleichen Sie die Eintragungen bitte jeweils mit der Auflistung der DRUCK-Routine, die im Anschluß an die INIT-Routine am Ende des Hex-Dump-Programms auf der Seite L19 aufgelistet ist.

In der Routine wird außer dem Akkumulator nur das C-Register in der CPU gebraucht. Mit dem ersten Befehl wird der ursprüngliche Inhalt des C-Registers (zusammen mit dem Inhalt des B-Registers) auf dem Stack hinterlegt. Gleich anschließend wird das ASCII-Byte aus dem Akkumulator im jetzt frei verfügbaren C-Register verwahrt.

Jetzt fragt die Routine beim Drucker an, ob ihm das zu druckende ASCII-Zeichen übergeben werden kann, oder ob er noch mit der Verarbeitung eines vorhergehenden Zeichens beschäftigt ist. Bei dieser Gelegenheit kann auch festgestellt werden, ob im Drucker Papier zum Drucken verfügbar ist.

Die Routine sieht sich sozusagen den derzeitigen Zustand des Drukkers an, den er über die Handshake-Signale Acknowledge, Busy und Paper Out an das System meldet. Diesen Zustand bezeichnen wir als STATus, und entsprechend beginnt die Abfrage des Status beim Label STAT der DRUCK-Routine.

Die Handshake-Signale liefert der Drucker als Bits Nr. 4, 5 und 6 an den Port B der Z 80 PIO. Von dort holt sich die Routine die drei Status-Signale mit einem IN A, (PIODB)-Befehl. Die darauf folgende Anweisung maskiert die drei interessierenden Bits (vgl. Lehrgang Mikroprozessortechnik, Seite S 66).

Bitte erinnern Sie sich: Der Drucker ist dann bereit, vom System ein ASCII-Byte zu übernehmen, wenn das Acknowledge-Signal den Wert 1 und die Signale Busy und Paper Out die Werte Ø haben (Seite H 27). Wenn es nur um die Signale Busy und Paper Out ginge, dann wäre leicht festzustellen, ob der Drucker bereit ist, ein ASCII-Byte zu übernehmen: Die Routine müßte nachsehen, ob nach der Maskierung alle vom Port B eingeholten Bits des Status-Bytes die Werte Ø haben.

Störend ist es, daß für die Übernahme eines Bytes das Acknowledge-Signal mit dem Wert 1 vorausgesetzt wird. – Im Bild H 33.1a haben wir oben das Bitmuster im Akkumulator dargestellt, das sich nach der Ausführung des IN A, (PIODB)-Befehls und der anschließenden Maskierung der (rot eingetragenen) Handshake-Signale ergibt, wenn der Drucker bereit ist, ein ASCII-Byte vom System zu übernehmen.

Der Programmablaufplan im Bild H 32.1 zeigt, daß nach der Maskierung der Handshake-Signale das Acknowledge-Signal invertiert wird. Im Bild H 33.1a sehen Sie, wie das durch die Exklusiv-ODER-Verknüpfung des Akkumulator-Inhalts mit dem Byte 1ØH (ØØØ1 ØØØØ) erreicht wird und welchen Zweck diese Maßnahme hat.

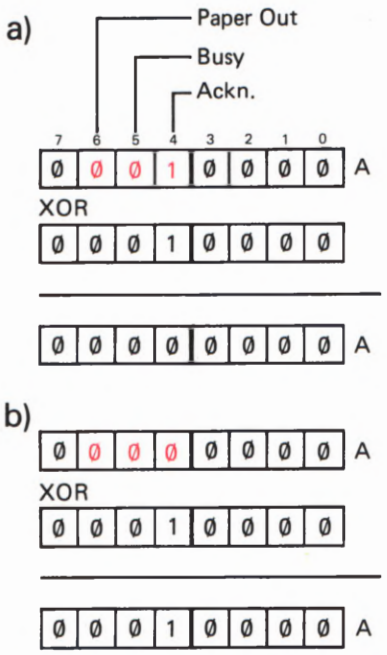


Bild H 33.1
Mit dem XOR 1Ø-Befehl wird das Ackn.-Bit Nr. 4 invertiert.
a) Mit den vom Drucker gelieferten Handshake-Signalen ergibt sich im Akku das Byte ØØ.
b) Jetzt ist das Byte im Akku ungleich ØØ: Der Drucker kann kein neues Zeichen empfangen.

Im Bild H 33.2 ist die Funktionstabelle für die Exklusiv-ODER-Verknüpfung von zwei Eingangssignalen e1 und e2 dargestellt. (Vgl. Lehrgang Mikroprozessortechnik ab der Seite S 56.) Sie erkennen, daß diese Verknüpfung immer dann das Ergebnis Ø liefert, wenn die beiden verknüpften Signale die gleichen Werte haben. Bei unterschiedlichen Werten der beiden verknüpften Signale ergibt sich der Wert 1.

e_1	e_2	a
Ø	Ø	Ø
Ø	1	1
1	Ø	1
1	1	Ø

Bild H 33.2
Funktionstabelle der Exklusiv-ODER-Verknüpfung.

Bei der Exklusiv-ODER-Verknüpfung eines Bytes im Akkumulator mit dem Operanden eines XOR-Befehls wird die Verknüpfung bitweise vorgenommen. Im Akkumulator ergeben sich an den einzelnen Bitstellen die Werte Ø, wenn das ursprüngliche Akkumulator-Bit und das zugehörige Bit im Operanden gleiche Werte haben. Nur bei unterschiedlichen Werten dieser Bits entsteht ein 1-Bit.

Das Bild H 33.1a zeigt, wie der Befehl XOR ØØØ1 ØØØØB (= XOR 1ØH) das Acknowledge-Bit invertiert. Für die dargestellten Werte der Handshake-Signale (die die Bereitschaft des Druckers für die Übernahme eines ASCII-Bytes signalisieren) ergibt sich nach der Ausführung des Exklusiv-ODER-Befehls im Akkumulator das Byte ØØ.

Im Teilbild b erkennen Sie, daß der XOR 1ØH-Befehl dann im Akkumulator ein von ØØ unterschiedliches Byte hinterläßt, wenn das Acknowledge-Signal den Wert Ø hat. In diesem Fall darf das System kein ASCII-Byte an den Drucker liefern. Die Abfrage des Drucker-

Status muß dann solange wiederholt werden, bis nach der Ausführung des XOR 1ØH-Befehls im Akkumulator das Byte ØØ erscheint. Der Befehl JR NZ,STAT bewirkt diese neuerliche Abfrage. (Vgl. die Programm-Liste auf der Seite L 19.)

Aufgabe H 34.1

Überlegen Sie bitte, welches Bitmuster sich im Akkumulator nach der Ausführung des XOR 1ØH-Befehls ergibt, wenn der Drucker mit den Handshake-Signalen Acknowledge = 1 und Busy = Ø die grundsätzliche Bereitschaft zur Übernahme eines weiteren ASCII-Bytes bekundet, wenn aber mit dem Signal Paper Out = 1 gemeldet wird, daß kein Papier zum Drucken mehr zur Verfügung steht.

Skizzieren Sie den Vorgang in einer Darstellung entsprechend dem Bild H 33.1.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 4.

Der weitere Ablauf der DRUCK-Routine ist denkbar einfach. Wenn das Programm festgestellt hat, daß der Drucker mit der Verarbeitung des vorhergehenden Zeichens FERTIG ist (Bild H 32.1), dann wird das anfangs im C-Register abgelegte ASCII-Zeichen wieder in den Akkumulator geholt und mit einem OUT (PIODA),A-Befehl über die PIO an den Drucker geschickt.

Auf der Seite H 25 haben wir erläutert, daß der Drucker frühestens 0,5 µs nach der Bereitstellung der Datenbits des ASCII-Bytes das *active Low*-STROBE-Signal vom System erwartet. (Vgl. Bild H 24.1.) In das Programm wird dazu eine Verzögerung (Bild H 32.1) eingebaut, die von zwei aufeinanderfolgenden EX (SP),HL-Befehlen bewirkt wird. Diese Befehle vertauschen jeweils den Inhalt des HL-Registerpaares mit dem Inhalt der vom Stackpointer angesprochenen Speicherzellen. Nach der Ausführung der beiden Befehle hat sich im System nichts geändert. Weil der Befehl EX (SP),HL aber eine recht lange Zeit für seine Ausführung gebraucht, eignet er sich gut für eine kurze Verzögerung.

Die anschließende Ausgabe des STROBE-Ø-Signals als Bit Nr. 1 über den entsprechenden Anschluß des PIO-Ports B und der gleich danach ausgegebene Wert 1 des gleichen Bits ist problemlos (vgl. Seite L 19).

Am Ende der Routine wird das ASCII-Byte wieder aus dem C-Register in den Akkumulator geholt und danach der ursprüngliche Inhalt des C-Registers wieder hergestellt. Nach dem abschließenden RET-Befehl findet das Hauptprogramm wieder die gleichen Register-Inhalte vor, die es vor dem Aufruf der DRUCK-Routine hergestellt hatte.

Interrupt-Programmierung der Z 80 CPU

Im Zusammenhang mit dem Thema dieses Lehrgangs ist bei der Interrupt-Programmierung naturgemäß die Programmierung der dabei eingesetzten Peripherie-Bausteine von vorrangigem Interesse. Wir haben Ihnen jedoch bereits auf der Seite S 25 gezeigt, daß beim Entwurf entsprechender Programme auch die CPU für die Verarbeitung von Interrupts eingerichtet (programmiert) werden muß.

Wir wollen Ihnen hier in erster Linie den Teil der Interrupt-Programmierung des Z 80 Mikroprozessors vorstellen, der im Zusammenhang mit den Fähigkeiten der Peripherie-Bausteine von Interesse ist. Einige Teile des Interrupt-Systems werden wir nur kurz der Vollständigkeit halber erwähnen.

Die Interrupt-Modes des Z 80

Der Z 80 CPU-Baustein hat zwei Interrupt-Anschlüsse. Den einen dieser Anschlüsse, der mit $\overline{\text{INT}}$ bezeichnet ist, haben wir bereits erwähnt (Seite S 22 und Bild H 19.1). Dieser Anschluß ist beim Arbeiten mit Interrupts, die über einen Peripherie-Baustein ausgelöst werden, von alleinigem Interesse. Er soll uns auch gleich noch ein wenig eingehender beschäftigen.

Um alle Möglichkeiten des Z 80 Interrupt-Systems wenigstens andeutungsweise zu erwähnen, wollen wir auch auf den zweiten Interrupt-Anschluß hinweisen. Er hat die Bezeichnung **NMI** als Abkürzung von **N***on* **M***askable* **I***nterrupt* (sprich: non maskebel interrupt). Ein Ø-Signal an diesem Eingang löst auf jeden Fall die Unterbrechung eines gerade laufenden Programms aus, unabhängig davon, ob die CPU durch einen EI-Befehl zur Annahme normaler Interrupt-Anforderungen programmiert worden ist oder nicht. Diese Eigenschaft wird durch die Bezeichnung „nicht maskierbar" beschrieben.

Eine Interrupt-Anforderung über den NMI-Anschluß hat vor allen anderen Interrupt-Anforderungen die höchste Priorität. Das bedeutet, daß ein Programm auch dann unterbrochen wird, wenn es gerade eine beliebige andere Interrupt-Service-Routine abarbeitet.

Die Z 80 CPU ist so eingerichtet, daß die Startadresse einer über den NMI-Anschluß aufgerufenen Interrupt-Service-Routine bei der Adresse ØØ66H liegen muß. In diesem Speicherbereich ist in Mikroprozessor-Systemen häufig das Betriebsprogramm untergebracht, das man nicht gern durch ein mitten hinein programmiertes Unterprogramm mit zwangsweise festgelegter Startadresse unübersichtlich macht. Wenn man vom NMI-Interrupt Gebrauch macht, dann programmiert man ab der Adresse ØØ66H nur einen Sprung zur tatsächlichen Startadresse der Interrupt-Service-Routine. Die drei Bytes dieses Sprungbefehls müssen dann von dem hier residierenden Programm wohl oder übel übersprungen werden. – Das Bild H 35.1 macht diese nur scheinbar komplizierte Programmierung deutlich.

Nicht maskierbare Interrupts werden meist Notfällen vorbehalten. Ein solcher Notfall liegt z. B. vor, wenn die Versorgungsspannung eines Systems zusammenbricht und mit der restlichen Ladung der

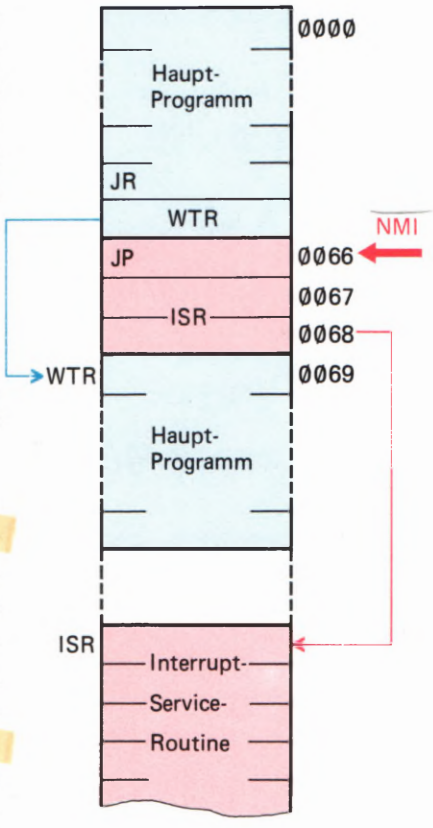


Bild H 35.1
Der Sprungbefehl zu einer NMI-Interrupt-Service-Routine bei der Adresse ØØ66H muß von einem normalen Programm übersprungen werden.

Siebkondensatoren rasch noch wichtige Inhalte des Schreib-Lese-Speichers in einen nicht flüchtigen Speicher gerettet werden sollen. Für den normalen Interrupt-Betrieb verwendet man den INT-Anschluß der CPU.

Über den **INT**-Anschluß können Interrupts auf drei unterschiedliche Arten bearbeitet werden. Diese Arten werden als **Modes** bezeichnet. Die CPU kann durch spezielle Befehle so programmiert werden, daß sie normale Interrupts in einem bestimmten der drei möglichen Modes Ø, 1 oder 2 bearbeitet.

Wenn die CPU für den Interrupt-**Mode Ø** programmiert worden ist, dann muß die Startadresse der Interrupt-Service-Routine als HOB grundsätzlich den Wert ØØ haben. Die Startadresse muß also im Bereich der Adressen ØØØØ und ØØFF liegen. Die CPU erwartet, daß die den Interrupt anfordernde Peripherie gleichzeitig mit der Anforderung den Operationscode eines der folgenden Befehle auf den Datenbus schaltet:

Befehl	Mnemon.	Opcode	Befehl	Mnemon.	Opcode
Restart ØØØØ	RST ØØ	C7	Restart ØØ2Ø	RST 2Ø	E7
Restart ØØØ8	RST Ø8	CF	Restart ØØ28	RST 28	EF
Restart ØØ1Ø	RST 1Ø	D7	Restart ØØ3Ø	RST 3Ø	F7
Restart ØØ18	RST 18	DF	Restart ØØ38	RST 38	FF

Restart bedeutet, daß die CPU eine Befehlsfolge startet, die bei der zusätzlich angegebenen Adresse beginnt. Bei der Ausführung der RST-Befehle legt die CPU automatisch den vorher aktuellen Inhalt des Programmzählers auf dem Stack ab. Wenn am Ende der Befehlsfolge, die bei der Restart-Adresse beginnt, ein RET-Befehl programmiert wird, dann arbeitet die mit RST aufgerufene Befehlsfolge wie ein normales Unterprogramm.

Die Restart-Befehle können im Gegensatz zu den JP- und JR-Befehlen mit einem einzigen Byte programmiert werden. Diese Vereinfachung ist möglich, weil das HOB der Sprungziel-Adresse bei den RST-Befehlen zwangsweise immer den Wert ØØ hat.

Im System der Speicheradressen eines Mikroprozessor-Systems bezeichnet man den Speicherbereich von jeweils 256 Speicherzellen, in dem die Adressen-HOBs gleiche Werte haben, als eine **Seite.** Wenn in einem Speicher die letzte Speicherzelle die Adresse FFFFH hat, dann gibt es in diesem Speicher die Seiten von ØØ bis FFH, also 256 Seiten. Die Seite ØØ wird in der Fachsprache auch als *Zero Page* (sprich: sirou päidsch) bezeichnet. In dieser Zero Page liegen die Zieladressen der Restart-Befehle.

Bei einem Mode-Ø-Interrupt hat die Peripherie die Möglichkeit, zwischen acht unterschiedlichen Interrupt-Service-Routinen auszuwählen, indem sie zusammen mit der Interrupt-Anforderung den Operationscode eines RST-Befehls auf den Datenbus schaltet. Die CPU holt sich in diesem Fall ihren Befehl nicht wie sonst üblich aus dem Speicher. Sie findet ihn ohne Speicherzugriff gleich auf dem Datenbus vor.

Ähnlich wie bereits beim NMI-Interrupt beschrieben, programmiert man üblicherweise ab der Restart-Adresse nicht die komplette Interrupt-Service-Routine. Dazu wäre bis zur nächsten Restart-Adresse

auch gar nicht der Platz. Bei der Restart-Adresse wird meist nur ein Sprung zum tatsächlichen Beginn der Interrupt-Service-Routine irgendwo im Speicher programmiert.

Der Interrupt-**Mode 1** hat Ähnlichkeit mit einem NMI-Interrupt: Es wird grundsätzlich ein Restart bei der Adresse ØØ38H ausgeführt. Auch bei einem in diesem Mode angefordertem Interrupt wird der aktuelle Inhalt des Programmzählers auf dem Stack abgelegt. Die jetzt angesprungene Befehlsfolge ab der Adresse ØØ38H funktioniert also als normale Interrupt-Service-Routine.

Der Unterschied dieses Interrupt-Modes zu einem NMI-Interrupt besteht darin, daß sich angeforderte Interrupts in eine Prioritätsstruktur einordnen müssen. Wir kommen darauf noch zurück.

Für Interrupts, die von einem Peripherie-Baustein angefordert werden, ist ausschließlich der **Mode 2** zuständig. In diesem Mode wird ein Interrupt-Vektor auf die niedrigere der Adressen von zwei Ablage-Speicherzellen gerichtet, in denen die Startadresse der Interrupt-Service-Routine abgelegt ist. Diese Methode haben wir bereits ab der Seite S22 eingehend beschrieben.

Das HOB des Interrupt-Vektors wird im I-Register der CPU abgelegt (— wie es da hineinkommt, wird nachher beschrieben —); das LOB des Interrupt-Vektors liefert der Peripherie-Baustein.

Wir wollen Ihnen jetzt noch die drei Befehle vorstellen, mit denen die CPU für einen bestimmten Interrupt-Mode programmiert wird:

Befehl	Mnemon.	Opcode
Interrupt-Mode Ø	IM Ø	ED 46
Interrupt-Mode 1	IM 1	ED 56
Interrupt-Mode 2	IM 2	ED 5E

Das I-Register

Voraussetzung für das Arbeiten der CPU im Interrupt-Mode 2 ist es, daß das HOB des Interrupt-Vektors im I-Register abgelegt ist. Das zugehörige LOB muß der CPU gleichzeitig mit der Interrupt-Anforderung geliefert werden.

Weil die CPU nur ein einziges I-Register besitzt, ergibt sich für den Interrupt-Mode 2 eine ähnliche Konsequenz, wie wir sie beim Mode Ø beschrieben haben. Dort mußten die Startadressen der Interrupt-Service-Routine sämtlich bei Adressen liegen, deren HOB den Wert ØØ hat. Im Mode 2 ist man nicht auf einen Speicherbereich angewiesen, dessen Adressen sämtlich ein HOB = ØØ haben. Auch brauchen in diesem Speicherbereich nicht die Startadressen der Interrupt-Service-Routinen selbst zu liegen, sondern nur die Ablage-Speicherzellen für diese Adressen. Gemeinsam ist jedoch beiden Modes, daß der Speicherbereich durch ein festgelegtes HOB begrenzt ist.

Im Interrupt-Mode 2 kann das HOB durch die Eintragung in das I-Register frei bestimmt werden, Wenn das aber einmal geschehen ist,

dann müssen alle Ablage-Speicherzellen für die Startadressen der Interrupt-Service-Routinen dieses im I-Register festgelegte HOB haben.

Der Befehlssatz des Z 80 Mikroprozessors enthält keinen Befehl, mit dem das I-Register – ähnlich wie z.B. der Akkumulator mit dem Befehl LD A,Byte – direkt mit einem bestimmten Byte geladen werden kann. Die einzige Möglichkeit zum Laden des I-Registers besteht auf dem Umweg über den Akkumulator: Das für das I-Register bestimmte HOB der Ablage-Adresse für die Startadresse der Interrupt-Service-Routine muß zunächst mit dem Befehl LD A,HOB in den Akkumulator gebracht werden. Von dort wird das HOB dann mit dem Befehl LD I,A in das I-Register kopiert.

Für das Laden des I-Registers ergibt sich damit folgende, aus vier Bytes bestehende Befehlsfolge:

```
LD A,HIGH ABLADR ;HOB der ABLage-ADResse in den Akkum.
LD I,A           ;HOB vom Akkumulator in das I-Register
```

Der Befehl LD I,A hat die Operationscodes ED 47. Es handelt sich also um einen Zwei-Byte-Befehl.

Interrupt-Prioritäten

Die Bearbeitung von Interrupt-Anforderungen der Peripherie ist meist eine eilige Angelegenheit. Peripherie-Signale können in vielen Fällen zu ganz unvorhersehbaren Zeiten eintreffen und dann nur sehr kurz anstehen. Es ist oft unmöglich, solche Signale per Programm mit Sicherheit zu erfassen und im Ablauf des Programms zu berücksichtigen. Die als Lösung der Aufgabe S 20.1 programmierte Änderung des Lauflicht-Programms hat Ihnen diese Tatsache recht deutlich vor Augen geführt. Die Interrupt-Programmierung bietet eine sehr elegante Lösung dieses Problems.

Bei der Vorstellung des Interrupt-Modes Ø haben wir darauf hingewiesen, daß die Peripherie die Möglichkeit hat, zwischen acht unterschiedlichen Interrupt-Service-Routinen auszuwählen (Seite H 36). Das ist natürlich nur dann sinnvoll, wenn die Peripherie mehrere verschiedene Interrupt-Signale liefert, die unterschiedliche Aktivitäten des Programms auslösen sollen.

Uns interessieren in diesem Lehrgang vorzugsweise Interrupt-Anforderungen, die von einem Peripherie-Baustein geliefert werden. – Sehen Sie sich bitte noch einmal das Bild S 22.1 an! In dieses Bild haben wir nur einen einzigen (über den PIO-Port A generierten) Interrupt-Vektor eingetragen. Offensichtlich kann auch der PIO-Port B einen solchen Interrupt-Vektor generieren, der dann auf eine Ablage-Speicherzelle mit der Startadresse für eine weitere Interrupt-Service-Routine zeigt.

Wir werden Ihnen noch Programme vorstellen, in denen mehrere, unterschiedliche Interrupts verarbeitet werden. Hier kommt es uns zunächst einfach auf die Tatsache an, daß solche unterschiedlichen Interrupts tatsächlich möglich sind.

Nun ist es durchaus denkbar, daß von der Peripherie ein Interrupt angefordert wird, wenn die CPU gerade die Service-Routine eines kurz vorher angeforderten, anderen Interrupts bearbeitet. Wie verhält sich das System in einem solchen Fall?

Es müssen zwei verschiedene Möglichkeiten unterschieden werden. Bei programmiertem Interrupt-Mode Ø liefert die Peripherie ihre Interrupt-Anforderung direkt – ohne Vermittlung über einen Peripherie-Baustein – an die CPU. Die CPU ist also selbst dafür verantwortlich, wie sie auf eine weitere Interrupt-Anforderung reagiert, wenn bereits eine Interrupt-Service-Routine in Arbeit ist.

Wenn im Interrupt-Mode 2 irgendwelche Peripherie-Bausteine des Z 80-Systems im Spiel sind, dann sorgen diese Bausteine dafür, ob eine Interrupt-Service-Routine ungestört ablaufen kann, oder ob diese Routine ihrerseits durch eine weitere Interrupt-Anforderung unterbrochen werden kann.

Hier liegt das Prinzip einer **Interrupt-Hierarchie,** also einer Interrupt-Rangordnung. Wir kommen gleich darauf zurück. Zunächst wollen wir zusehen, wie die CPU mit einer Interrupt-Anforderung fertig wird, die während des Ablaufs einer Interrupt-Service-Routine eintrifft.

Die CPU enthält intern ein spezielles **Interrupt-Flipflop IFF1,** das sich nach dem erstmaligen Einschalten der CPU bzw. nach einem RESET im Ruhezustand befindet, also zurückgesetzt ist. (Es gibt noch ein Interrupt-Flipflop IFF2, das uns aber in diesem Zusammenhang nicht zu interessieren braucht.) Bei diesem Zustand des Flipflops IFF1 verweigert die CPU die Bearbeitung von Interrupt-Anforderungen.

Die einzige Möglichkeit, das Flipflop IFF1 zu setzen, ist die Ausführung des bereits auf der Seite S22 vorgestellten Befehls EI (*Enable Interrupts*). Nur wenn das geschehen ist, kann die CPU maskierbare Interrupts, die über den Anschluß INT angefordert werden, (im Gegensatz zu nicht maskierbaren Interrupts über den NMI-Anschluß, vgl. Seite H35) berücksichtigen.

> Wenn die CPU nach der Annahme einer Interrupt-Anforderung eine Interrupt-Service-Routine aufgerufen hat, wird automatisch das Flipflop IFF1 zurückgesetzt. Weitere Interrupt-Anforderungen werden erst dann wieder berücksichtigt, wenn das Flipflop IFF1 mit dem Befehl EI gesetzt worden ist.

Diese Eigenschaft der CPU bringt es mit sich, daß in Arbeit befindliche Interrupt-Service-Routinen zunächst grundsätzlich nicht durch weitere Interrupts unterbrochen werden können. Es ist deshalb aber auch unbedingt notwendig, am Ende einer Interrupt-Service-Routine immer einen EI-Befehl zu programmieren. Wenn dieser Befehl vergessen wird, dann kann das System nach dem Start eines Programms nur ein einziges Mal einen Interrupt ausführen; für weitere Interrupts bleibt die CPU gesperrt.

Sie können sich das sehr gut in einem Versuch ansehen:

Versuch H40.1

Vergessener EI-Befehl

Laden Sie Ihr System mit dem Lauflicht-Programm, das ab der Seite L23 aufgelistet ist (vgl. den Versuch S19.1).

Am Ende der ALARM-Routine haben wir als vorletzten Befehl bei der Adresse 1853H das Byte FB für den Befehl EI programmiert. Setzen Sie anstelle dieses Befehls bei der Adresse 1853 einen NOP-Befehl mit dem Byte ØØ ein und starten Sie das Programm! Unterbrechen Sie das Programm durch die Betätigung der zum Port A gehörenden Taste Nr. 0 oder Nr. 1. Die ALARM-Routine wird pflichtgemäß ausgeführt; gleichzeitig mit dem Aufruf dieser Routine setzt die CPU intern das Flipflop IFF1 zurück. Das können Sie natürlich nicht sehen. Die Auswirkung dieses Zurücksetzens merken Sie aber sofort, wenn Sie einen zweiten Interrupt auslösen wollen: Ab sofort verweigert die CPU die Annahme weiterer Interrupts.

Aufgabe H40.1

Warum kann die CPU bei der Ausführung des Versuchs H40.1 der ersten Interrupt-Anforderung nachkommen? — Wie können Sie die Ausführung von Interrupts von vornherein verhindern?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü6.

Im Lauflicht-Programm haben wir den EI-Befehl am Ende der Interrupt-Service-Routine ALARM vor dem RETI-Befehl programmiert, der diese Routine abschließt. Dadurch bleibt das IFF1-Flipflop vom Beginn bis zum Ende der Routine gesperrt. Die Routine kann ihrerseits nicht durch weitere Interrupts unterbrochen werden.

Selbstverständlich besteht die Möglichkeit, den EI-Befehl gleich zu Anfang der Interrupt-Service-Routine zu programmieren. In diesem Fall wird das IFF1-Flipflop zwar beim Aufruf der Routine zurückgesetzt, aber sofort anschließend wieder gesetzt. Das bedeutet, daß die CPU auch während der Bearbeitung der Interrupt-Service-Routine für Interrupt-Anforderungen empfänglich ist: Die Routine kann ihrerseits von Interrupts unterbrochen werden.

Der hier beschriebene Mechanismus läßt sich oft sehr nutzbringend auch innerhalb des Hauptprogramms verwenden. Wenn der Ablauf des Hauptprogramms in bestimmten Teilen zeitkritisch ist, dann darf die Ablaufzeit dieses Programmteils nicht durch aufgerufene Interrupt-Service-Routinen unterbrochen werden. Man kann einen solchen Aufruf einfach dadurch verhindern, daß man zu Anfang des betreffenden Programmteils das Interrupt-Flipflop IFF1 per Befehl zurücksetzt und es erst dann wieder setzt, wenn der zeitkritische Programmteil abgearbeitet ist.

Zum Rücksetzen des Flipflops IFF1 wird der auf der Seite S22 vorgestellte D*isable* I*nterrupt*-Befehl DI verwendet. Der EI-Befehl macht die CPU wieder für Interrupts empfänglich.

Wir wollen uns jetzt ansehen, wie die Sache mit der Interrupt-Hierarchie im Interrupt-Mode 2 funktioniert, wenn mit Peripherie-Bausteinen gearbeitet wird.

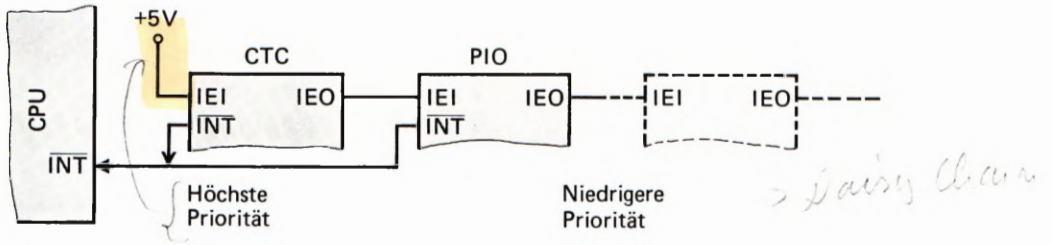


Bild H41.1
Im Micro-Professor-System sind der CTC-Baustein und die PIO in einer *Daisy Chain* angeordnet, welche die Interrupt-Hierarchie bestimmt.

Im Micro-Professor-System werden zwei Interface-Bausteine der Z 80-Familie verwendet. Das Bild H 19.1 zeigt den Anschluß der PIO an die CPU. Etwas versteckt ist im unteren Teil des Bildes auch der CTC-Baustein angedeutet. Wir mußten diesen Baustein bei der Darstellung der PIO bereits mit berücksichtigen, weil der Anschluß IEI der PIO direkt zu einem Anschluß IEO am CTC führt.

Im Bild H41.1 haben wir das Prinzip des interessierenden Teils aus dem Bild H 19.1 noch einmal dargestellt. Sie erkennen, daß sowohl die PIO als auch der CTC-Baustein (und auch alle anderen Peripherie-Bausteine des Z 80-Systems) jeweils einen IEI- und einen IEO-Anschluß haben. Sind in einem System mehrere Peripherie-Bausteine eingesetzt, dann werden diese Bausteine über IEI- und IEO-Anschlüsse schaltungsmäßig miteinander „verkettet". Eine solche Schaltung wird im englischen mit dem blumigen Namen *Daisy Chain* bezeichnet. Wörtlich heißt das: Gänseblümchen-Kette. Weil da nun schon eine Gans drin vorkommt, möchten wir diese Bezeichnung wohl mit „Gänsemarsch-Schaltung" übersetzen.

Die (im Bild H41.1 links beginnende) Reihenfolge der verketteten Peripherie-Bausteine bestimmt durch die Hardware-Anordnung die Interrupt-Hierarchie. Bei dem in der Reihenfolge als erstem angeordneten Peripherie-Baustein wird der IEI-Anschluß (*Interrupt Enable In:* Interrupt-Freigabe-Eingang) auf HIGH-Potential gelegt. Damit wird dieser Peripherie-Baustein grundsätzlich ermächtigt, Interrupt-Anforderung der Peripherie an die CPU weiterzugeben. (Voraussetzung dafür ist natürlich, daß auch die Programmierung des Bausteins diese Weitergabe zuläßt.)

Wenn der Baustein eine Interrupt-Anforderung an die CPU geschickt hat, dann quittiert die CPU die Annahme dieser Anforderung durch eine Kombination der Signale IORQ und M1 (vgl. das Bild H 19.1). Als Folge dieser Quittung schaltet der Baustein das Signal an seinem IEO-Anschluß (*Interrupt Enable Out:* Interrupt-Freigabe-Ausgang) auf LOW-Potential.

Das jetzt auf LOW-Potential liegende IEO-Signal des ersten Peripherie-Bausteins wird dem in der *Daisy Chain* nächsten Peripherie-Baustein als IEI-Signal zugeführt. Dieser nächste Baustein ist jetzt solange für die Weitergabe von Interrupt-Anforderungen gesperrt, wie sein IEI-Eingang auf LOW-Potential liegt. Und weil er nun schon selbst gesperrt ist, schaltet er seinerseits seinen IEO-Anschluß auf LOW-Potential.

Es ist leicht einzusehen, daß damit auch der in der Kette dritte Baustein (wenn vorhanden) für die Weitergabe von Interrupt-Anforderungen gesperrt ist.

Hinsichtlich der IEI- und IEO-Anschlüsse verhalten sich alle Peripherie-Bausteine des Z 80-Systems gleich. Daraus ergibt sich diese Folgerung: Wenn ein Peripherie-Baustein eine Interrupt-Anforderung mit Erfolg an die CPU weitergegeben hat, dann sind alle in der *Daisy Chain* weiter rechts stehenden Bausteine für die Weitergabe von Interrupt-Anforderungen gesperrt.

Aufgabe H 42.1

a) Überlegen Sie bitte, welche Werte die IEI- und IEO-Signale in der Anordnung entsprechend dem Bild H 41.1 haben, wenn über die PIO eine Interrupt-Anforderung an die CPU weitergegeben wurde und von dieser quittiert wurde.

b) Was geschieht, wenn während der Bearbeitung der Interrupt-Service-Routine für die PIO der CTC eine Interrupt-Anforderung an die CPU weitergeben will?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 6.

Unsere Überlegungen zeigen im Zusammenhang mit der Darstellung im Bild H 41.1, was es mit der Interrupt-Hierarchie auf sich hat. In der Hierarchie hat der Peripherie-Baustein die höchste Interrupt-Priorität, der in der *Daisy Chain* am weitesten links steht. Wenn es die Programmierung des Befehls EI in der Interrupt-Service-Routine zuläßt, dann kann ein Peripherie-Baustein mit höherer Priorität mit einer Interrupt-Anforderung auch dann zu Wort kommen, wenn ein Baustein mit niedrigerer Priorität bereits einen Interrupt ausgelöst hat. Umgekehrt muß ein Peripherie-Baustein mit niedrigerer Priorität mit seiner Interrupt-Anforderung stets warten, bis der Interrupt eines Bausteins mit höherer Priorität vom Programm bedient worden ist.

Wir haben auf der Seite H 41 festgestellt, daß ein Peripherie-Baustein solange für die Weitergabe von Interrupt-Anforderungen gesperrt ist, wie ihm an seinen IEI-Anschluß vom IEO-Anschluß des vorhergehenden Bausteins ein Ø-Signal geliefert wird. Es wird ihm erst dann wieder ein 1-Signal an den IEI-Anschluß geliefert, wenn die vom vorhergehenden Baustein aufgerufene Interrupt-Service-Routine abgearbeitet worden ist.

Jetzt erhebt sich natürlich sofort die Frage, woher denn ein Peripherie-Baustein wissen soll, daß die CPU mit der Bearbeitung der von ihm angeforderten Interrupt-Service-Routine fertig ist. Der Baustein muß das unbedingt erfahren, denn sonst kann er an seinen IEO-Anschluß kein 1-Signal mehr schalten. Ohne dieses 1-Signal würde kein Baustein mit niedrigerer Priorität mehr mit Erfolg einen Interrupt anfordern können.

Eine Interrupt-Service-Routine ist im Prinzip nichts anderes als ein normales Unterprogramm. Der Interrupt-Mechanismus der CPU sorgt dafür, daß beim Aufruf dieses Interrupt-Unterprogramms der aktuelle Inhalt des Programmzählers auf dem Stack abgelegt wird; am Ende des Unterprogramms sorgt ein Return-Befehl für den Rücksprung an die richtige Stelle des Hauptprogramms.

Das ist genau die gleiche Funktion, die auch bei einem normalen Unterprogramm abläuft. Im Unterschied zu einem normalen Unter-

programm muß in einer Interrupt-Service-Routine das Interrupt-Flipflop mit einem EI-Befehl wieder gesetzt werden. Sonst besteht scheinbar kein Unterschied. Oder doch? – Es besteht sogar ein ganz gewichtiger Unterschied.

Ein normales Unterprogramm wird mit einem RET-Befehl abgeschlossen, der die Funktion eines POP PC-Befehls hat: Er sorgt dafür, daß der Programmzähler zur Fortsetzung des Programms mit der Adresse geladen wird, die vorher beim Aufruf des Unterprogramms auf dem Stack abgelegt worden ist. (Vgl. Lehrgang Mikroprozessortechnik, Seite H72.) Mit der Ausführung dieses Befehls ist die Bearbeitung des Unterprogramms für die CPU erledigt.

Anders ist es bei der Bearbeitung einer Interrupt-Service-Routine. Der diese Routine abschließende Befehl muß zunächst die gleiche Funktion haben wie ein RET-Befehl: Er muß durch die Funktion pop pc dafür sorgen, daß das Programm an der Stelle fortgesetzt wird, an der es durch die Interrupt-Anforderung unterbrochen wurde. Er muß aber zusätzlich noch eine weitere Funktion erfüllen: An den Peripherie-Baustein, der die Interrupt-Anforderung übermittelt hat, muß die Information geliefert werden, daß jetzt die Bearbeitung der Interrupt-Service-Routine abgeschlossen ist. Denn nur diese Information macht es möglich, daß der Peripherie-Baustein seinen IEO-Anschluß wieder auf HIGH-Potential schaltet.

Wir stellen Ihnen hier drei Befehle zusammen, die im Prinzip die gleichen Funktionen ausüben: Das Zurückspringen des Programms aus einer aufgerufenen Subroutine:

Befehl	Mnem.	Opcode
Return von Unterprogramm	RET	C9
Return von mask. Interrupt	RETI	ED 4D
Return von nicht mask. Interrupt	RETN	ED 45

Der erste dieser Befehle ist der Ihnen bereits hinreichend bekannte RET-Befehl zum Abschluß eines normalen Unterprogramms.

Der RETI-Befehl erfüllt genau die zusätzliche Funktion, die wir gerade vorher beschrieben haben: Beim Rücksprung von einer Interrupt-Service-Routine erkennt die PIO den RETI-Befehl und schließt daraus, daß die Interrupt-Service-Routine abgeschlossen ist. Sie schaltet daraufhin wieder ein 1-Signal an ihrem IEO-Anschluß und gibt die Peripherie-Bausteine mit niedrigerer Interrupt-Priorität für die Weitergabe von Interrupt-Anforderungen frei.

Bemerkenswert ist noch, daß der Peripherie-Baustein erst dann wieder in der Lage ist, eine neue Interrupt-Anforderung an die CPU weiterzugeben, wenn er einen RETI-Befehl erkannt und daraus geschlossen hat, daß eine zuvor aufgerufene Interrupt-Service-Routine abgeschlossen ist.

Den Befehl RETN haben wir nur der Vollständigkeit halber aufgeführt. Er hat eine dem RETI-Befehl entsprechende Funktion und muß am Ende einer Interrupt-Service-Routine programmiert werden, die von einem nicht maskierten Interrupt aufgerufen wird (s. S. H35).

Versuch H 44.1

Die Wirkung des RETI-Befehls

Laden Sie bitte noch einmal das Lauflicht-Programm (Seite L 23) in Ihr System! – Ersetzen Sie das erste Byte (ED) des RETI-Befehls am Ende der ALARM-Routine bei der Adresse 1854H durch den Operationscode C9 des RET-Befehls!

Starten Sie das Programm, und unterbrechen Sie es dann durch die Betätigung einer der zum Port A gehörenden Tasten Nr. 0 oder Nr. 1.

Der durch diese Betätigung angeforderte Interrupt wird normal ausgeführt. Warum auch nicht? Das Interrupt-Flipflop IFF1 in der CPU ist durch den Befehl EI in der INIT-Routine gesetzt worden, und zur Zeit der Interrupt-Anforderung wurde keine Interrupt-Service-Routine bearbeitet. Am IEI-Anschluß der PIO liegt also ein 1-Signal.

Versuchen Sie nach Ablauf der soeben ausgelösten Interrupt-Service-Routine einen zweiten Interrupt auszulösen! – Warum funktioniert das nicht?

Das Programm ist aus der Interrupt-Service-Routine richtig zurückgekehrt; dafür hat der jetzt wirksame RET-Befehl gesorgt. Sie erkennen es daran, daß das Lauflicht ordnungsgemäß weiter läuft.

Der RET-Befehl hat aber nicht dafür gesorgt, daß der PIO das Ende der Interrupt-Service-Routine mitgeteilt wird.

Die PIO schaltet an ihren IEO-Anschluß kein 1-Signal. Das merken Sie in ihrem Versuch nicht, denn im Micro-Professor-System ist der PIO in der *Daisy Chain* kein weiterer Peripherie-Baustein nachgeschaltet. Die PIO kann aber keine weitere Interrupt-Anforderung an die CPU übermitteln, solange sie nicht aus einem RETI-Befehl das Ende der Interrupt-Service-Routine erkannt hat. Und ein solcher Befehl fehlt jetzt im Programm.

Halten Sie das Programm durch die Betätigung der RESET-Taste an und ersetzen Sie das Byte C9 des RET-Befehls bei der Adresse 1854H wieder durch das erste Byte ED des RETI-Befehls! Jetzt hat das Programm wieder seine ursprüngliche Form. Starten Sie das Programm und betätigen Sie eine der Interrupt-Tasten! – Warum funktioniert die Sache jetzt immer noch nicht?

Es kann ja auch gar nicht gehen. Die PIO hat noch keinen RETI-Befehl erkannt, und solange das nicht geschehen ist, muß sie annehmen, daß die vorher aufgerufene Interrupt-Service-Routine noch immer nicht abgeschlossen ist.

Sorgen Sie dafür, daß die CPU den RETI-Befehl ausführt. Sie müssen die CPU sozusagen mit Gewalt dazu zwingen, die Interrupt-Service-Routine ablaufen zu lassen. Per Interrupt kommt sie ja nicht hin.

Betätigen Sie die RESET-Taste RS und starten Sie das Programm bei der Anfangsadresse 183B der ALARM-Routine! Jetzt wird der Alarm ausgeführt. Auch der RETI-Befehl wird abgearbeitet, aber Ihr System quittiert das mit der Anzeige SYS- SP. Kümmern Sie sich nicht darum. Machen Sie RESET und starten Sie das Programm erneut. Und siehe da: Die Sache klappt wieder.

Centronics-Schnittstelle mit Interrupt

Im Lehrbrief 2 haben wir uns ausführlich mit einer sogenannten parallelen Schnittstelle für den Anschluß eines Druckers befaßt und dabei bereits angekündigt, daß sich ein solcher Anschluß wesentlich eleganter herstellen läßt, wenn man von den Möglichkeiten der Interrupt-Programmierung Gebrauch macht.

Wir wollen Ihnen hier zwei Programme für eine Centronics-Schnittstelle zeigen, die mit Interrupts arbeiten. Im ersten dieser Programme wollen wir ganz „brav" vorgehen und trotz des Interrupts recht konventionell arbeiten. Im zweiten Programm zeigen wir Ihnen dann, wie einfach die Sache ist, wenn man die Möglichkeiten der Z 80 PIO voll ausnutzt.

Interrupt in der Betriebsart 3

Unseren Überlegungen wollen wir das gleiche Programm zugrunde legen, das wir auch im Lehrbrief 2 verwendet haben: Ein Programm, mit dem der Inhalt eines Speicherbereichs als Hex-Dump ausgedruckt wird (vgl. die Seiten H 30 und L 17). Wie in diesem Programm der Ausdruck formatiert wird, wie also an den Anfang jeder Zeile die vier Ziffern für eine Adresse kommen und wie anschließend sechzehn Bytes aus dem Speicher mit jeweils einem Abstand zum Druck bereitgestellt werden, soll uns weiter nicht interessieren. Sie können das gegebenenfalls in der Programm-Auflistung nachlesen.

Interessant ist hier allein die DRUCK-Routine, die ein ASCII-Byte aus dem Akkumulator über die Z 80 PIO an den Drucker überträgt. Damit diese Übertragung in der gewünschten Weise funktioniert, muß die PIO entsprechend programmiert werden, und das geschieht mit einer Initialisierungs-Routine INIT.

In unserem Programm beginnt die DRUCK-Routine bei der Adresse 1865H (vgl. Seite L 20). Gleich im Anschluß an die DRUCK-Routine beginnt die Initialisierungs-Routine INIT. Den Entwurf dieser beiden Routinen wollen wir uns jetzt für den Fall überlegen, daß dabei von den Möglichkeiten des Interrupts Gebrauch gemacht wird.

Lesen Sie bitte noch einmal den Abschnitt über die Centronics-Druck-Routine ab der Seite H 32 nach! Wir haben dort erläutert, daß die CPU vor dem Aussenden des ASCII-Bytes aus dem Akkumulator jeweils nachsehen muß, ob der Drucker bereit ist, dieses Zeichen zu übernehmen. Solange das nicht der Fall ist, muß das Zeichen im Akkumulator verwahrt werden. Es darf erst dann abgeschickt werden, wenn der Drucker mit den Signalen Busy = Ø, Acknowledge = 1 und Paper Out = Ø meldet, daß er das vorher übermittelte Byte verarbeitet hat und außerdem noch hinreichend Papier zum Drucken vorhanden ist.

Diese vom Drucker gelieferten Signale haben wir als Drucker-Status bezeichnet. Das Bild H 32.1 zeigt, wie in der im Lehrbrief 2 beschriebenen Druck-Routine (vgl. Seite L 20) der Drucker-Status in einer beim Label STAT beginnenden Schleife abgefragt wird. Die Schleife

wird solange immer von neuem durchlaufen, bis die Status-Signale erkennen lassen, daß die CPU ihr ASCII-Byte abschicken darf.

Unsere Überlegungen zeigen, daß sich an dieser Stelle die Verwendung eines Interrupts fast von selbst anbietet. Der Drucker meldet zu einem von der CPU nicht vorauszusehenden Zeitpunkt seine Bereitschaft, das jeweils nächstfolgende ASCII-Byte anzunehmen. Auf diese Bereitschaft muß die CPU mehr oder weniger untätig warten. Ganz untätig kann sie nicht sein, denn sie muß ja den Drucker in der Abfrage-Schleife immer wieder fragen, ob sie jetzt ihr Byte endlich loswerden kann.

Viel geschickter wäre es doch sicher, wenn sich die CPU um den Drukker überhaupt nicht zu kümmern brauchte und einer nützlicheren Tätigkeit nachgehen könnte. Wenn der Drucker mit der Verarbeitung des vorher angelieferten ASCII-Bytes fertig ist, dann kann er der CPU seine Bereitschaft zur Entgegennahme des nächsten ASCII-Bytes über ein Interrupt-Signal mitteilen. Die CPU unterbricht daraufhin kurz ihre nützliche Tätigkeit und liefert das ASCII-Byte ab. Sie fährt dann in ihrer normalen Tätigkeit fort, bis der Drucker mit einem neuerlichen Interrupt das nächstfolgende ASCII-Byte anfordert.

Das Bild H 46.1 zeigt, wie der Drucker an die PIO angeschlossen werden kann, wenn von einer Interrupt-Meldung des Druckers Gebrauch gemacht werden soll. Vergleichen Sie die Darstellung mit der im Bild H 25.1! Zunächst wird Ihnen auffallen, daß auf die Verwendung des Handshake-Signals Acknowledge verzichtet wurde. Das kann ohne Bedenken geschehen: Das Busy-Signal allein genügt, um der CPU mitzuteilen, wann der Drucker mit der Verarbeitung des vorhergehenden ASCII-Bytes fertig ist. Solange das Busy-Signal den Wert 1 hat, ist der Drucker noch mit der Verarbeitung eines angelieferten Bytes beschäftigt (vgl. Seite H 25); wenn das Busy-Signal den Wert Ø annimmt, darf das nächstfolgende ASCII-Byte angeliefert werden.

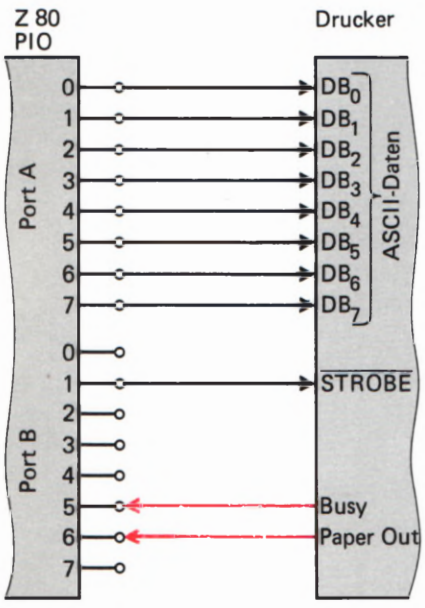


Bild H 46.1
Anschluß eines Druckers mit Centronics-Schnittstelle bei Verwendung einer Interrupt-Service-Routine für die Übergabe von Zeichen an den Drucker.

Jetzt ist sehr einfach zu formulieren, wann die CPU ein ASCII-Byte an den Drucker ausliefern kann: Der Drucker muß mit der Verarbeitung des vorhergehenden ASCII-Bytes fertig sein (Busy = Ø) **UND** zu diesem Zeitpunkt muß im Drucker noch genügend Papier verfügbar sein. (Paper Out = Ø). Die Kombination dieser beiden Signale löst ein Interrupt-Signal für die CPU zum Ausliefern eines ASCII-Bytes aus.

Die so formulierte Aufgabe ist mit Ihren inzwischen erworbenen Kenntnissen über die Programmierung der PIO leicht zu lösen. Es soll zunächst nur festgestellt werden, daß die vom Interrupt aufgerufene Interrupt-Service-Routine die Anweisung enthalten muß, ein (ASCII-)Byte aus dem Akkumulator an den Port A der PIO zu schicken. Außerdem muß sie Anweisungen zur Aussendung eines STROBE-Signals enthalten, mit der das vorher ausgesandte ASCII-Byte gültig gemacht wird (vgl. Seite H 25).

Wir werden die Programmierung der PIO zur Lösung dieser Aufgabe und die Programmierung der Interrupt-Service-Routine gleich noch kurz erläutern.

Zunächst wollen wir feststellen, daß die CPU bei einer solchen Anwendung eines Interrupts viel Zeit gewinnt. Sie braucht sich mit der Abfrage-Schleife jetzt nicht mehr aufzuhalten.

Die Frage ist, mit welcher nützlichen Tätigkeit wir die CPU während der nun frei verfügbaren Zeit beschäftigen wollen. Ganz sicher würde uns da bei längerem Nachdenken etwas Gescheites einfallen. Man könnte zum Beispiel die Bereitstellung der Adressen am Anfang der Hex-Dump-Zeilen und das Abholen der auszudruckenden Bytes aus dem Speicher in diese Zeit verlegen.

Wir wollen es uns ganz einfach machen: Wir schicken die CPU während des Wartens auf den Interrupt in Urlaub. Die eben als Beispiel genannte, nützliche Verwendung der freigewordenen Zeit würde eine Umstrukturierung unseres ganzen Hex-Dump-Programms erfordern. Und der Erfolg wäre in unserem speziellen Fall recht fragwürdig, denn den Drucker können wir ja doch nicht zu schnellerem Arbeiten bewegen. Die CPU hätte ihre frei verfügbare Zeit dann eben an anderer Stelle.

In unserem Programm ist der Ausdruck des Hex-Dumps alleinige Aufgabe. Mehr wollen wir gar nicht. Es ist aber durchaus denkbar, daß die CPU ein ganz anderes Programm erledigen muß, in dem zwischendurch das Ausdrucken von ASCII-Bytes als lästige Nebenaufgabe anfällt. In einem solchen Fall ist man natürlich froh, wenn man die CPU nicht immer wieder in die zeitraubende Abfrage-Schleife nach dem Drucker-Status schicken muß.

Wie geht das nun aber mit dem CPU-Urlaub? Für diesen Fall stellt der Befehlssatz des Z 80 Mikroprozessors einen besonderen Befehl zur Verfügung:

Befehl	Mnem. Code	Opcode
Warten auf Interrupt	HALT	F3 76 H

Nach einem HALT-Befehl stellt die CPU ihre Tätigkeit ein. Sie führt intern lauter NOP-Befehle aus (damit sie nicht ganz einschläft) und wartet auf das Eintreffen eines Interrupts oder eines Hardware-Resets. Wenn eine Interrupt-Anforderung eintrifft, dann nimmt die CPU ihre normale Tätigkeit sofort wieder auf und springt die Interrupt-Service-Routine an, nachdem sie vorher in gewohnter Weise den aktuellen Inhalt des Programmzählers auf dem Stack abgelegt hat.

Wir werden unser Hex-Dump-Programm wie folgt ablaufen lassen:

Die Druck-Routine ist ein normales Unterprogramm, der ein im Akkumulator bereitgestelltes ASCII-Byte übergeben wird. Statt nun aber – wie es bisher geschehen ist – in einer Warte-Schleife auf den passenden Drucker-Status zu warten, liefert die Routine das ASCII-Byte ohne weitere Formalitäten sofort über den A-Port der PIO an den Drucker. Es wird also einfach vorausgesetzt, daß der Drucker für die Übernahme des Bytes bereit ist. Gleich anschließend wird das Byte durch ein STROBE-Signal gültig gemacht.

Danach wird die CPU mit einem HALT-Befehl in Urlaub geschickt. Sie wartet jetzt auf eine Interrupt-Anforderung vom Drucker, die dann kommt, wenn der Drucker nicht mehr damit beschäftigt (busy) ist, das angelieferte Byte zu verarbeiten. Außerdem muß natürlich hinreichend Papier zum Drucken vorhanden sein.

Was muß die Interrupt-Anforderung bewirken? Im hier vorliegenden Fall eigentlich gar nichts, außer daß sie die CPU aus ihrem Urlaub zurückholt. Die CPU kann dann gleich den auf den HALT-Befehl in der Druck-Routine folgenden Befehl ausführen. Und das ist ein RET-Befehl, der den Rücksprung aus der Druck-Routine in das Hauptprogramm bewirkt. Dort wird das nächste ASCII-Byte im Akkumulator bereitgestellt und dann die Druck-Routine von neuem aufgerufen. Das gerade beschriebene Spiel kann sich jetzt wiederholen.

Die Interrupt-Service-Routine kann in unserem speziellen Fall also geradezu lächerlich einfach sein. Weil sie ja keine andere Aufgabe hat, als die CPU wieder an ihre normale Arbeit zu bringen (von der sie mit dem HALT-Befehl beurlaubt wurde), muß sie lediglich mit einem EI-Befehl dafür sorgen, daß das Interrupt-Flipflop IFF1 wieder gesetzt wird. Sie erinnern sich: Dieses CPU-interne Flipflop wird nach einer akzeptierten Interrupt-Anforderung zurückgesetzt. Die CPU kann erst dann eine weitere Interrupt-Anforderung akzeptieren, wenn IFF1 wieder gesetzt ist. (Vgl. Seite H39.)

Außer dem EI-Befehl gehört zu unserer Interrupt-Service-Routine nur noch der RETI-Befehl, der den Rücksprung aus der Routine bewirkt (Seite H43).

Bitte beachten Sie: Normalerweise wird eine Interrupt-Service-Routine selbstverständlich weitaus umfangreichere Aktivitäten entfalten als in unserem Beispiel. Unsere merkwürdige Programmierung kommt nur daher, daß wir im Hex-Dump-Programm mit der gewonnenen Zeit beim besten Willen nichts besseres anzufangen wissen.

Aufgabe H48.1

Für das Hex-Dump-Programm mit Interrupt muß die Z 80 PIO teilweise anders programmiert werden als in dem im Lehrbrief 2 ab der Seite L17 aufgelisteten Programm.

Schreiben Sie bitte eine Initialisierungs-Routine INIT für die PIO mit folgender Programmierung (vgl. Bild H46.1):

Port A: Betriebsart 3.
Alle Anschlüsse sind Ausgänge.

Port B: Betriebsart 3
Die Anschlüsse 5 und 6 sind Eingänge, alle anderen Anschlüsse sind Ausgänge.
Der Interrupt-Vektor zeigt auf die Adresse mit dem Label VWAIT. (Bei dieser und der folgenden Adresse ist die Anfangsadresse der Interrupt-Service-Routine abgelegt.) Das LOB dieser Ablage-Adresse (LOW VWAIT) wird der PIO übergeben.
Dem *Interrupt Control Word* folgt eine Interrupt-Maske. Zu beachten sind die Signalwerte der Busy- und Paper Out-

Signale, die eine Interrupt-Anforderung auslösen sollen. – Diese Anforderung soll nur dann ausgelöst werden, wenn Busy und Paper Out **gleichzeitig** die richtigen Signalwerte haben. – Die PIO soll gleich nach der Programmierung in der Lage sein, Interrupt-Anforderungen an die CPU weiterzugeben.
Die Port-Anschlüsse für die Signale Busy und Paper Out sind Interrupt-Eingänge.

Nehmen Sie für die Programmierung die Tafel T 3 zu Hilfe. – Sie brauchen nur das Quellprogramm *(Source Program)*, also die mnemonischen Codes und die Operanden der Befehle anzuschreiben.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü7.

Ab der Seite L39 haben wir die Druck-Routine und die zugehörige Initialisierungs-Routine aufgelistet. Diese bei der Adresse 1865H beginnenden Routinen können die ebenfalls bei der Adresse 1865H beginnenden Druck- und Initialisierungs-Routinen ab der Seite L20 ersetzen.

Wenn anstelle der alten Druck- und Initialisierungs-Routinen die entsprechenden Routinen mit Interrupt-Funktion eingesetzt werden, dann ist zu beachten, daß jetzt die INIT-Routine nicht mehr bei der Adresse 187F beginnt, sondern bei der Adresse 1871H. Am Anfang des Hex-Dump-Programms muß also bei der Adresse 18Ø1H das Byte 7F in das Byte 71H für den richtigen Aufruf der INIT-Routine geändert werden.

Unwichtig für die Funktion, aber im Interesse der Vollständigkeit ist es, daß sich die Endadresse des Programms ändert. Bei der Adresse 18Ø7H sollte also anstelle des Bytes 8F das Byte 9A eingesetzt werden.

Sehen Sie sich die neuen Routinen bitte an! Es wird Ihnen sofort auffallen, daß die Druck-Routine beim Arbeiten mit Interrupt deutlich kürzer geworden ist. Die am meisten ins Auge fallende Vereinfachung ergibt sich durch den Wegfall der Abfrage-Schleife für den Drucker-Status. Zusätzlich wurde darauf verzichtet, das zu druckende ASCII-Byte nach seiner Ablieferung an den Drucker noch im Akkumulator zur Verfügung zu haben. Und schließlich erweist es sich als nicht unbedingt notwendig, zwischen der Ablieferung des Bytes an den Drucker und der Aussendung des STROBE-Signals eine zusätzliche Verzögerung zu programmieren. Die beiden EX (SP),HL-Befehle können also weggelassen werden.

Länger geworden ist die Initialisierungs-Routine. Das ist verständlich: Dem PIO-Port B müssen für die Interrupt-Programmierung zusätzlich zum *Mode-* und *I/O Register Control Word* drei weitere Befehle übermittelt werden. Außerdem muß die CPU für den Interrupt-Betrieb programmiert werden. Dieser Aufwand ist natürlich bei dem relativ kurzen Hex-Dump-Programm besonders auffällig. Bei umfangreicheren Programmen fällt der Aufwand für die Interrupt-Programmierung der PIO nicht mehr ins Gewicht.

Die Interrupt-Programmierung bringt eindeutig einen Zeitgewinn für die CPU, auch wenn dieser Zeitgewinn im hier betrachteten Sonderfall nicht genutzt wird.

Interrupt in der Betriebsart Ø

Sicher ist die Interrupt-Programmierung einer Centronics-Schnittstelle in der Betriebsart 3 recht interessant, aber so richtig überzeugend ist der Gewinn gegenüber der konventionellen Programmierung doch eigentlich nicht.

Lesen Sie bitte noch einmal zur Erinnerung den Abschnitt über die Betriebsarten der Z 80 PIO ab der Seite S2 durch! In der Betriebsart Ø arbeiten die acht Port-Anschlüsse der PIO ausschließlich als Sender – und das ist doch eigentlich genau das richtige für die Übermittlung eines ASCII-Bytes an den Drucker. Wenn man bei der bisherigen Programmierung der PIO den Port A für die Betriebsart Ø einrichtet, dann könnte man sich zumindest das *I/O Register Control Word* sparen, also vier Bytes in der Initialisierungs-Routine.

Die Programmierung des Ports B für die Betriebsart 3 scheint zunächst unumgänglich zu sein, denn der Handshake-Betrieb spielt sich in beiden Signal-Richtungen ab.

Sehen Sie sich bitte das Bild S3.1 an, in dem die Port-Anschlüsse so dargestellt sind, wie sie in der Betriebsart Ø arbeiten. Vielleicht fällt es Ihnen jetzt erst auf, weil es bei der Beschäftigung mit der Betriebsart 3 nicht von Interesse war: Zu jedem Port gehören außer den acht Daten-Anschlüssen noch zwei weitere Anschlüsse, von denen der eine grundsätzlich als Eingang arbeitet und der andere als Ausgang. Diese beiden Anschlüsse (READY und $\overline{\text{STROBE}}$) sind eigens für einen Handshake-Betrieb vorgesehen, wie Sie ihn z.B. beim Anschluß des Druckers kennengelernt haben.

Die Funktion der beiden Handshake-Anschlüsse haben wir bereits auf der Seite S3 beschrieben. Wir wollen das hier noch einmal wiederholen:

Sobald an den acht Daten-Anschlüssen ein Byte bereit steht, das von der Peripherie übernommen werden kann, schaltet die PIO auf den zum Port gehörenden **READY**-Anschluß ein 1-Signal. Dieses 1-Signal kann die Peripherie veranlassen, das bereitgestellte Byte abzuholen.

Wenn die Peripherie das Byte vom Port abgeholt hat, kann sie ihrerseits diese Tatsache mit einem kurzen Ø-Signal an den mit **$\overline{\text{STROBE}}$** bezeichneten PIO-Anschluß melden. Die Ø-1-Flanke am Ende des Ø-Signals löscht das vorher ausgesandte 1-Signal am READY-Anschluß der PIO. Dadurch wird vermieden, daß die Peripherie das gleiche Byte zweimal liest.

Diese Funktionsbeschreibung der Handshake-Anschlüsse an der PIO läßt bereits vermuten, daß eine fast ideale Anschluß-Möglichkeit für einen Drucker mit Centronics-Schnittstelle gegeben ist. Sie werden gleich noch einen kleinen Schönheitsfehler feststellen, aber auch wirklich nur einen ganz kleinen.

Zunächst ist anzumerken, daß die Bezeichnung der PIO-Anschlüsse im Zusammenhang mit den Centronics-Anschlüssen des Druckers etwas unglücklich ist – ein Problem, auf das schon auf der Seite H25 hingewiesen wurde. Sowohl an der PIO als auch am Drucker befindet sich ein mit $\overline{\text{STROBE}}$ bezeichneter Anschluß, aber diese beiden Anschlüsse gehören keineswegs zusammen.

Im Bild H51.1a ist dargestellt, wie der Drucker bei Verwendung der Betriebsart Ø an einen PIO-Port angeschlossen wird.

Sehen Sie sich zunächst den STROBE-Anschluß des Druckers an. Über diesen Anschluß wird dem Drucker mit einem LOW-Signal (1-Ø-Signalwechsel, vgl. Seite H25) mitgeteilt, daß die an seine Daten-Anschlüsse gelieferten Signale jetzt gültig sind. Ein entsprechendes Signal liefert die PIO an ihrem READY-Anschluß. Aber – und hier steckt der kleine Schönheitsfehler – dieses READY-Signal hat genau die falsche „Polarität": Die Bits an den Daten-Anschlüssen der PIO werden mit einem HIGH-Signal (Ø-1-Signalwechsel, vgl. Seite H50) gültig gemacht.

Normalerweise regelt der Fachmann solche kleinen Unstimmigkeiten selbstverständlich per Software. Dieser Weg ist hier leider versperrt, denn der Drucker ist ein in sich abgeschlossenes, fertig gekauftes Gebilde, dem man nur selten bastelnd zu Leibe rücken mag. Bei der PIO ist ein Eingriff erst recht unmöglich: Der von ihr gelieferte Signalwert ist durch ein internes Mikroprogramm festgelegt.

Das Bild H51.1a zeigt, wie man sich helfen kann: Zwischen dem READY-Anschluß der PIO und dem STROBE-Anschluß des Drukkers wird ein NICHT-Glied geschaltet.

Klaglos dagegen funktioniert die Sache mit dem vom Drucker gelieferten Acknowledge-Signal: Der Drucker bestätigt die Verarbeitung eines angelieferten Bytes mit einem kurzzeitigen LOW-Signal (1-Ø-Signalwechsel, Seite H25). Im Gegensatz zum Busy-Signal, das *active High* ist (Seite H25), und das bei der vorhergehenden Programmierung verwendet wurde (Seite H46), hat dieses vorher verschmähte Acknowledge-Signal gerade die „Polarität", die der STROBE-Anschluß der PIO für die Empfangsbestätigung des angelieferten Bytes benötigt.

Im Bild H51.1b ist der zeitliche Zusammenhang zwischen dem von der PIO gelieferten READY-Signal und dem Quittungs-Signal Acknowledge vom Drucker dargestellt. Die End-Ø-1-Flanke des Acknowledge-Signals löscht das READY-Signal der PIO. Der Drukker kann also nicht zweimal dasselbe Byte von der PIO abholen.

Unsere Darstellung zeigt, daß bei dieser Anschlußart auf die Auswertung des Paper Out-Signals vom Drucker verzichtet werden muß. Dafür ist aber die Software jetzt ebensowenig aufwendig wie die Hardware. Sehen Sie sich bitte die DRUCK- und INIT-Routinen ab der Seite L43 an!

Der grundsätzliche Aufbau des Programms entspricht genau den Überlegungen, die wir zur vorhergehenden Programmierung angestellt haben: Die Druck-Routine setzt voraus, daß der Drucker zur Übernahme des angelieferten Bytes bereit ist und schickt dieses Byte sofort und ungefragt an den Drucker. Danach wartet die CPU im HALT-Zustand auf eine Interrupt-Anforderung vom Drucker. Sie kommt dann, wenn der Drucker das soeben angelieferte Byte verarbeitet hat.

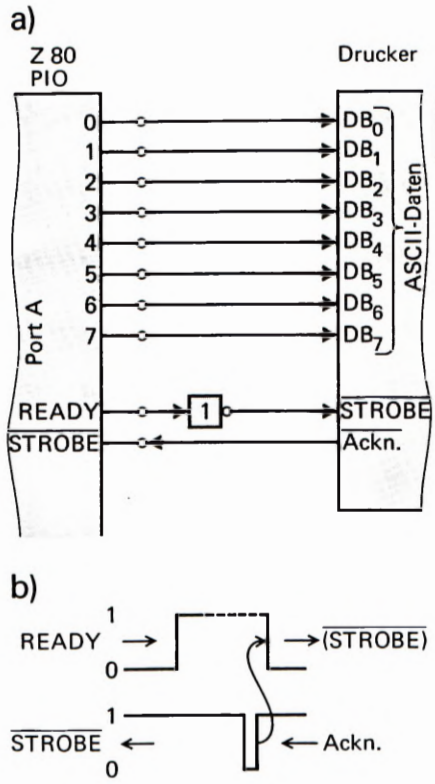


Bild H51.1
a) Beim Anschluß eines Druckers mit Centronics-Schnittstelle muß in der PIO-Betriebsart Ø das READY-Signal invertiert werden.
b) Das Acknowledge-Signal des Druckers löscht das READY-Signal der PIO.

Die Interrupt-Service-Routine besteht wieder nur aus den beiden Befehlen EI und RETI. Sie schickt das Programm zum RET-Befehl am Ende der DRUCK-Routine. Jetzt kann das Hauptprogramm das nächstfolgende Byte zum Druck bereitstellen.

Vergleichen Sie die DRUCK-Routine mit der DRUCK-Routine auf der Seite L 39! Es fällt Ihnen sofort auf, daß jetzt die Befehle zur Generierung des STROBE-Signals fehlen. Dieses STROBE-Signal liefert der READY-Anschluß des PIO-Ports A automatisch.

Am vorteilhaftesten wirkt sich die Verwendung der Betriebsart Ø bei der PIO-Programmierung in der INIT-Routine aus. Das läßt bereits die Aufstellung auf der Seite T 3 erkennen: Für diese Betriebsart braucht im Prinzip dem *Mode Control Word* nur noch ein *Interrupt Vector Word* zu folgen.

In der Praxis stellt sich heraus, daß die PIO ausdrücklich ermächtigt werden muß, Interrupt-Anforderungen an die CPU weiterzugeben. Bei der Programmierung der Betriebsart 3 geschieht das mit dem Wert 1 des Bits Nr. 7 im *Interrupt Control Word.* Da dieser Befehl bei der Programmierung der Betriebsart Ø fehlt, muß die Interrupt-Freigabe mit dem Wert 1 des Bits Nr. 7 in einem *Interrupt Disable Word* erfolgen.

Damit ist die Programmierung der PIO dann aber auch schon fertig. Die Interrupt-Programmierung der CPU wird durch die gewählte Betriebsart der PIO natürlich nicht betroffen; sie enthält immer die gleichen Befehle.

Aufgabe H 52.1

In dem auf der Seite L 43 aufgelisteten Programm ist das Vektorfeld mit den Ablage-Speicherzellen für die Anfangsadresse der Interrupt-Service-Routine gleich im Anschluß an die Interrupt-Service-Routine angelegt.

In umfangreicheren Programmen kann dieses Vektorfeld an ganz anderer Stelle liegen.

Nehmen Sie an, im Anschluß an die Routinen unseres Programms folgten noch eine ganze Reihe anderer Programm-Module, deren letztes Befehls-Byte bei der Adresse 1946H abgelegt ist. Erst nach diesem Byte soll das Vektorfeld angelegt werden.

Welche Änderungen müssen in dem ab der Seite L 43 aufgelisteten Programm vorgenommen werden, wenn das Vektorfeld im angegebenen Speicherbereich liegen soll?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 7.

Aufbau und Anschluß des Z 80 CTC

Den grundsätzlichen Aufbau des CTC mit seinen vier Kanälen und den für die Programmierung wichtigsten Anschlüssen haben Sie bereits im vorangegangenen Abschnitt kennengelernt. Im Bild S 46.1 ist dieser Aufbau schematisch dargestellt.

Wenn man die vielfältigen Möglichkeiten des CTC voll nutzen will, dann sind etwas genauere Kenntnisse des internen Aufbaus eines Kanals notwendig. Vor allem kann die Bedeutung der einzelnen Bits im *Channel Control Word* erst erkannt werden, wenn man Einzelheiten der Funktion eines Kanals durchschaut. Dieser Aufbau ist bei allen vier Kanälen der gleiche.

Wir wollen Ihnen in diesem Abschnitt den internen Aufbau eines Kanals des CTC vorstellen, soweit er für die effektive Programmierung von Interesse ist. Anschließend zeigen wir Ihnen am Beispiel des Micro-Professors, wie der CTC in ein Mikroprozessor-System eingebaut wird. Und schließlich ist es besonders interessant, wie sich der CTC in die Interrupt-Struktur des Z 80 Mikroprozessors einfügt.

Der interne Aufbau eines CTC-Kanals

Sehen Sie sich bitte das Bild H 55.1 an und vergleichen Sie es mit dem Bild S 46.1! Sie erkennen die Darstellung des Blockschaltplans eines Teils des CTC. Orientieren Sie sich bitte anhand der äußeren Anschlüsse, welchen Teil des CTC wir abgebildet haben. Sie finden den Kanal Ø mit den Ihnen schon bekannten Anschlüssen ZC/TO0 und CLK/TRG0, sowie die Interrupt-Leitung. Auf die Darstellung der Kanäle 1 bis 3 haben wir verzichtet. Bis auf den fehlenden Anschluß ZC/TO beim Kanal 3 ist die Schaltung dieser Kanäle identisch mit der des Kanals Ø.

Im linken Teil der Darstellung finden Sie zwei Funktionsblöcke, die für alle vier Kanäle gleichermaßen zuständig sind. Ihre Funktionen sind hier nicht weiter interessant; wir brauchen sie nicht näher zu erläutern.

Wichtig sind die internen Funktionen des Kanals selbst. Sein Herz stellt der 8-Bit-Abwärts-Zähler dar, der bei der Programmierung über das *Time Constant Register* (Zeitkonstanten-Register) mit dem *Time Constant Word* auf einen bestimmten Wert geladen wird.

Im Versuch S 46.1 haben Sie den Inhalt des Abwärts-Zählers mit einzelnen Peripherie-Impulsen dekrementiert. Bitte vergessen Sie zunächst einmal diesen Versuch! Vergessen Sie im Augenblick überhaupt, daß der CTC im *Counter Mode* betrieben werden kann, daß also der Inhalt des Abwärts-Zählers mit einzelnen Impulsen über den CLK/TRG-Anschluß dekrementiert werden kann! Die Bezeichnungen im Blockschaltplan beziehen sich sämtlich auf den *Timer Mode*, also auf das Arbeiten des CTC als Uhr, die vom Quarz-genauen System-Takt (Clock) synchronisiert wird.

Bevor dem CTC über den System-Datenbus Befehls-Bytes übermittelt werden, verhält er sich völlig neutral; er tut gar nichts. Auch nach der Programmierung mit einem *Channel Control Word* tut er noch nichts. Er weiß dann lediglich, was er zu tun hat, wenn seine Aktivitäten gestartet werden.

Bei der erstmaligen Programmierung muß dem CTC nach dem *Channel Control Word* unbedingt noch ein *Time Constant Word* geschickt werden, mit dem das *Time Constant Register* geladen wird. Dieses Register überträgt seinen Inhalt in den Abwärts-Zähler, wenn dessen Inhalt null ist. Von diesem Zustand kann bei der erstmaligen Programmierung ausgegangen werden.

Nach der Eintragung einer Zeitkonstanten hängt es von der Programmierung mit dem *Channel Control Word* ab, was geschieht. Entweder beginnt der CTC – gesteuert vom System-Takt (Clock) – sofort damit, den Inhalt des Abwärtszählers zu dekrementieren, oder er wartet auf ein kurzzeitiges 1-Signal von der Peripherie, mit dem das Abwärts-Zählen gestartet wird. Dieser Start des Abwärts-Zählens wird als **Triggern** bezeichnet. Das Peripherie-Trigger-Signal erhält der Kanal über den CLK/**TRG**-Anschluß, der von diesem **TRiG**gern seinen Namen hat.

Nach dem automatischen oder getriggerten Start des Abwärts-Zählens liefert der System-Takt (Clock) die Steuer-Impulse für den *Count Down*. Der 8-Bit-Abwärts-Zähler wird – beginnend beim voreingestellten Wert – bis zum Inhalt null leergezählt. Es wurde bereits erläutert, was geschieht, wenn die Uhr abgelaufen ist: Der Abwärts-Zähler wird mit dem im *Time Constant Register* abgelegten Wert nachgeladen; außerdem wird an den Anschluß ZC/TO ein kurzzeitiges 1-Signal geliefert. Die Bedeutung der Abkürzung ZC kennen Sie bereits (**Z***ero* **C***ount*, auf null gezählt). Mit TO wird *Time* **O***ut* abgekürzt: Die Zeit ist vorbei, die Uhr ist abgelaufen.

Zusätzlich zur Ausgabe des ZC/TO-Signals wird der Abwärts-Zähler sofort wieder aus dem *Time Constant Register* nachgeladen, und dann beginnt der Leerzähl-Zyklus wieder von vorn, unabhängig davon, ob das Leerzählen automatisch begonnen hat, oder ob es von der Peripherie getriggert wurde.

Sehen Sie sich bitte noch einmal die Bilder S 41.1 und S 43.2 an! In dem im Bild S 41.1 dargestellten Beispiel liefert der CTC an seinen Ausgang ZC/TO eine ununterbrochene Impulsfolge. Das wird bei der Programmierung für automatische Triggerung erreicht.

Das Bild S 43.2 zeigt ein Beispiel für die Verwendung des Eingangs CLK/TRG, an den die Peripherie Trigger-Impulse liefert. Der CTC liefert nach einer einstellbaren Verzögerung ein Echo des Triggerimpulses. Sie wissen jetzt, wie das funktioniert. Die Verzögerung ist um so länger, je größer der Wert ist, mit dem der Abwärts-Zähler aus dem *Time Constant Register* vorgeladen wird. Je mehr im Zähler drinsteht, um so länger dauert es, bis er leergezählt ist. Die in diesem Bild dargestellte Arbeitsweise des CTC erfordert spezielle Programm-Maßnahmen, die hier nicht weiter interessieren.

Nun interessiert Sie natürlich, wie lang die Leerzähl-Zeit denn nun wirklich ist. Dazu wollen wir eine kleine Rechnung aufmachen. Wir gehen zunächst davon aus, daß anfangs sämtliche Bits des Zählers mit

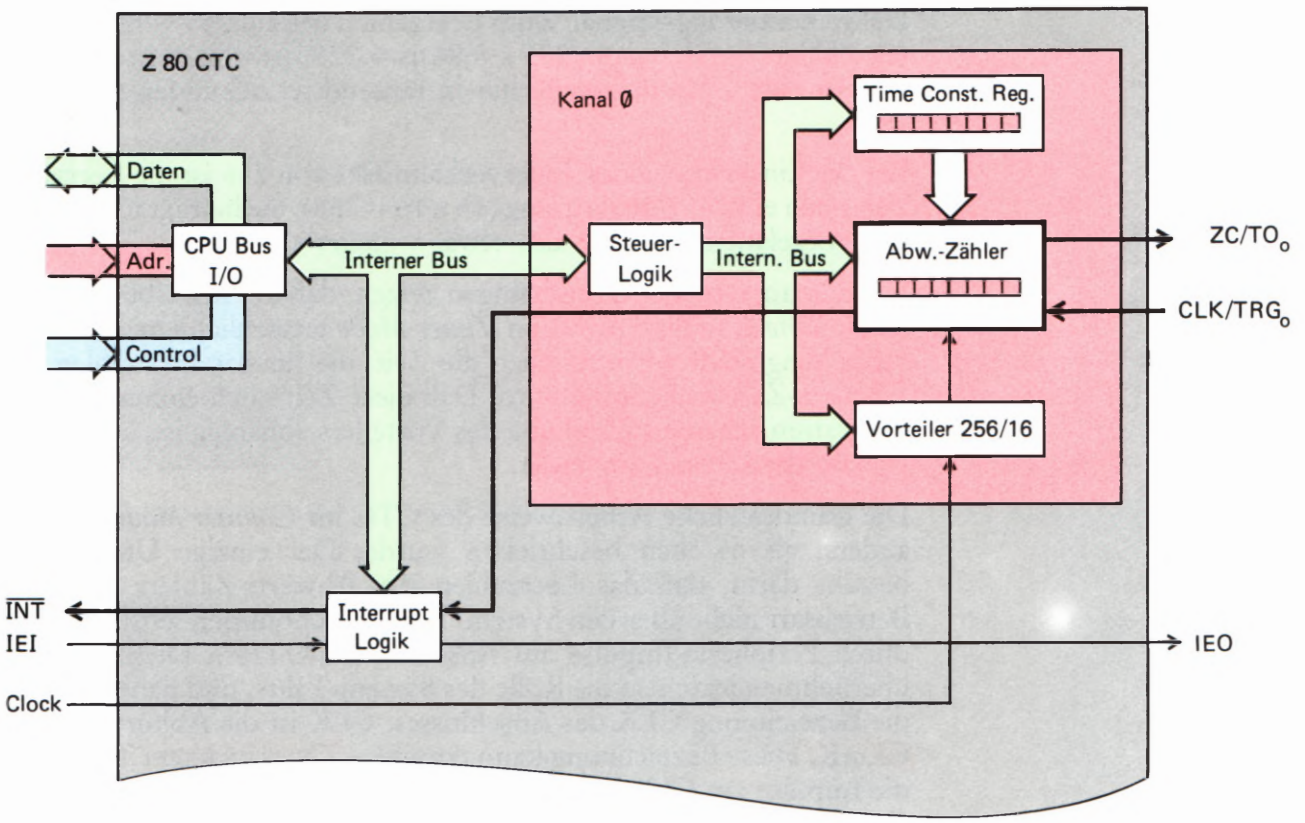


Bild H55.1
Interner Aufbau eines CTC-Kanals. Bis auf den fehlenden Anschluß ZC/TO beim Kanal 4 sind alle vier CTC-Kanäle gleich aufgebaut.

Werten 1 geladen sind. Der Zähler hat also den Inhalt 1111 1111B = FFH = 255.

Es wurde bereits gesagt, daß im *Timer Mode* der System-Takt die Dekrementierungs-Impulse für den Zähler liefert. Den im Bild H55.1 eingetragenen Vorteiler wollen wir im Augenblick einmal vergessen. In diesem Fall wird der Zähler-Inhalt mit jeder Periode der Taktfrequenz um eins erniedrigt.

Der Micro-Professor arbeitet mit einem Quarz, der ungefähr mit der Frequenz 3,58 MHz schwingt. Diese Frequenz wird elektronisch durch zwei geteilt und dann dem Z 80 CTC zugeführt. Die Taktfrequenz beträgt also etwa 1,79 MHz. Daraus ergibt sich eine Perioden-Dauer von 1/1,79 MHz = Ø,56 µs.

Wenn der Abwärts-Zähler im CTC alle Ø,56 µs um eins dekrementiert wird, dann ist der mit 255 vorgeladene Zähler in Ø,56 µs x 255 = 142,8 µs leergezählt. Für die Arbeitsgeschwindigkeit eines Mikroprozessors ist das zwar eine beachtliche Zeit, aber eine vielleicht gar mechanisch arbeitende Peripherie würde eine so kurze Verzögerung sicher nicht einmal wahrnehmen: Die Zeit zwischen zwei Leerzähl-Signalen am CTC-Anschluß müßte in millionstel Sekunden gemessen werden!

Damit merklichere Leerzähl-Zeiten erreicht werden können, wird im CTC nicht die Taktfrequenz selbst zum Dekrementieren des Zählers verwendet, sondern wahlweise die durch 16 oder durch 256 dividierte Taktfrequenz. Die Teilung bewirkt der im Bild H55.1 eingetragene Vorteiler, dessen Teilerverhältnis mit dem *Channel Control Word* bei der Programmierung des CTC eingestellt werden kann.

Wenn der Vorteiler auf ein Teilerverhältnis von 16 programmiert ist, dann erscheint am Abwärts-Zähler alle 16 x Ø,56 µs = 8,96 µs ein

Dekrementierungs-Signal. Zum Leerzählen des auf 255 voreingestellten Zählers werden dann 255 x 8,96 µs = 2285 µs = 2,29 ms benötigt. Das ist eine Zeit, die immerhin in tausendstel Sekunden gemessen wird.

Bei der Einstellung eines Teilerverhältnisses von 256 ist die Leerzähl-Zeit noch einmal 16mal so lang (16 x 16 = 256). Sie beträgt für den mit 255 vorgeladenen Zähler also etwas weniger als 37 ms.

Die hier angestellten Überlegungen zeigen, daß mit der Übermittlung eines *Time Constant Words* im *Timer Mode* tatsächlich eine Zeitkonstante eingestellt wird, nämlich die Zeit, die für das Leerzählen des Abwärts-Zählers benötigt wird. Daß diese Zeit noch einmal von der programmierbaren Einstellung des Vorteilers abhängig ist, ändert das Prinzip der Überlegung nicht.

Die grundsätzliche Arbeitsweise des CTC im *Counter Mode* ist nicht anders, als es eben beschrieben wurde. Der einzige Unterschied besteht darin, daß das Leerzählen des Abwärts-Zählers in dieser Betriebsart nicht über den System-Takt vorgenommen wird, sondern durch Peripherie-Impulse am Anschluß CLK/TRG. Diese Impulse übernehmen jetzt also die Rolle des System-Takts, und daher kommt die Bezeichnung **CLK** des Anschlusses: CLK ist die Abkürzung von **CL***oc***K.** Diese Bezeichnung kann täuschen. *Clock* bedeutet Takt, aber die Impulse am CLK/TRG-Anschluß brauchen keineswegs in einem gleichmäßigen Takt einzutreffen. Der Versuch S 46.1 hat das gezeigt.

Bei der Programmierung eines CTC-Kanals für den *Counter Mode* wird der Weg vom Clock-Anschluß des CTC zum Abwärts-Zähler des Kanals unterbrochen. Der CLK/TRG-Anschluß ändert seine Funktion: Er triggert jetzt nicht mehr den vom System-Takt gesteuerten Leerzähl-Zyklus, sondern übernimmt selbst die Dekrementierung des Zählers.

Nach wievielen Peripherie-Impulsen der Abwärts-Zähler jetzt leergezählt wird, hängt wieder davon ab, mit welchem Wert er aus dem *Time Constant Register* vorgeladen wird. Das *Time Constant Register* vermittelt also im *Counter Mode* tatsächlich keine Zeitkonstante mehr, sondern eine einfache Zähl-Vorgabe. Sie wird aber beim Programmieren des Kanals mit dem *Time Constant Word* eingestellt.

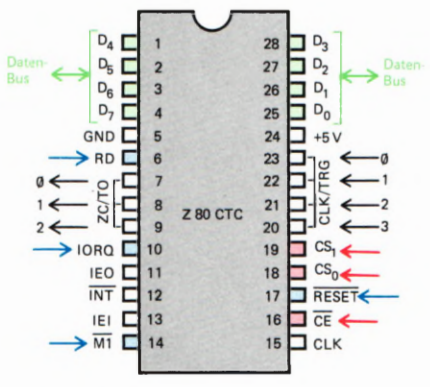


Bild H 56.1
Anschlüsse des Z 80 CTC-Bausteins.

Die Adressierung des Z 80 CTC

Das Bild H 55.1 zeigt den CTC so, wie ihn der Programmierer beim Anschreiben einer Initialisierungs-Routine und bei der Verwendung in einem Software-Konzept sehen muß. Er setzt selbstverständlich voraus, daß der Baustein funktionsmäßig richtig in das System eingefügt wurde und fragt den Hardware-Hersteller nur nach den Adressen, über die er die einzelnen Kanäle erreichen kann.

Beim Entwurf der Schaltung des Systems muß man sich natürlich Gedanken machen, wie der Daten- und der Steuerbus angeschlossen werden, und vor allem, wie die Adressierung der vier Kanäle realisiert werden kann.

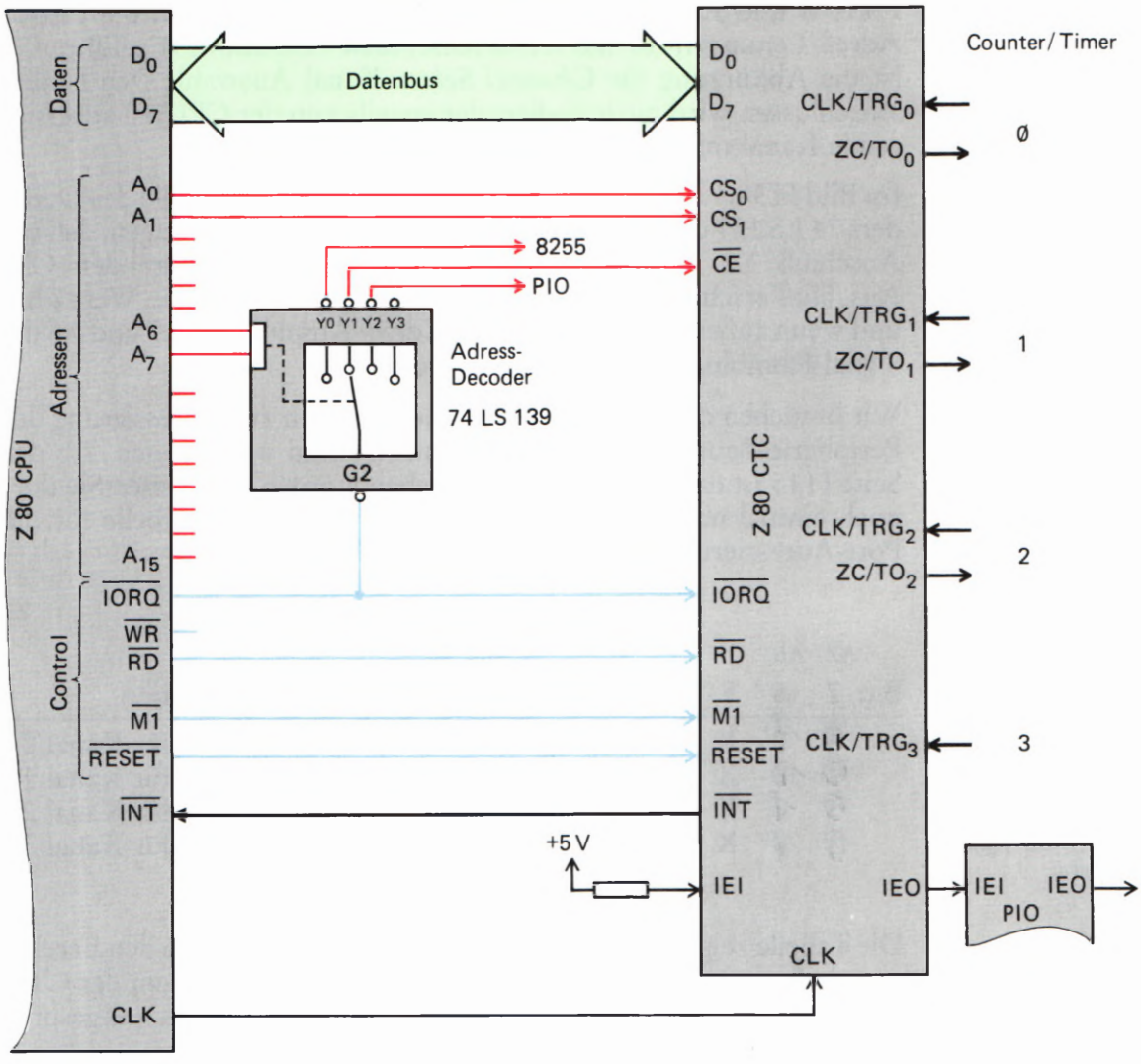


Bild H57.1
So ist der Z 80 CTC im Micro-Professor an die CPU angeschlossen.

Im Bild H56.1 haben wir der Vollständigkeit halber die Anschluß-Belegung des Z 80 CTC dargestellt. Viel interessanter ist der Schaltplan im Bild H57.1. Er zeigt, wie der CTC im Micro-Professor eingebaut ist.

Vergleichen Sie das Bild bitte mit dem Schaltplan für die Z 80 PIO im Bild H19.1! Sie müssen schon recht genau hinsehen, um die Unterschiede festzustellen. Das ist kein Wunder, denn sowohl die PIO als auch der CTC gehören zur Z 80-Familie. Selbstverständlich sind alle Mitglieder dieser Familie weitgehend nach den gleichen Gesichtspunkten aufgebaut, und deshalb sind auch die Anschlußpläne all dieser Bausteine nahezu identisch.

Abgesehen von den Ausgangs-Anschlüssen des CTC, die selbstverständlich anders aussehen als die der PIO, besteht der wesentliche Unterschied in der Beschaltung des Adreß-Decoders 74 LS 139. Der Anschluß $\overline{CE}$ (*Chip Enable* – Baustein-Aktivierung) der PIO ist mit dem Anschluß Y2 des Adreß-Decoders verbunden; der Anschluß CE des CTC ist an den Anschluß Y1 des Decoders geführt.

Unterschiedlich sind beim CTC und bei der PIO die Anschlüsse gekennzeichnet, die mit den Anschlüssen A0 und A1 der CPU verbunden sind. Diese Anschlüsse bewirken bei der PIO die Adressen-Auswahl für Befehle und Daten (C/$\overline{D}$) bzw. für die Ansprache der PIO-

Ports B und A (B/A, vgl. Seite H16). Beim CTC sind die beiden Adreß-Leitungen AØ und A1 an Anschlüsse CSØ und CS1 geführt. CS ist die Abkürzung für C*hannel* S*elect:* Kanal Auswahl. Den beiden Anschlüssen wird binär codiert der jeweils von der CPU zu adressierende Kanal mitgeteilt.

Im Bild H58.1 haben wir noch einmal die Funktionstabelle des Decoders 74 LS 139 dargestellt. Die farbig unterlegten Werte zeigen, daß am Anschluß Y2 nur dann ein Ø-Signal für den invertierenden CE-Anschluß erscheint, wenn das Signal am Anschluß G2 den Wert Ø hat und wenn außerdem von den CPU-Adreß-Anschlüssen A7 und A6 die Signal-Kombination 1Ø geliefert wird.

G2	A_7	A_6	YØ	Y1	Y2	Y3
1	X	X	1	1	1	1
Ø	Ø	Ø	Ø	1	1	1
Ø	Ø	1	1	Ø	1	1
Ø	1	Ø	1	1	Ø	1
Ø	1	1	1	1	1	Ø

Bild H58.1
Funktionstabelle des Bausteins 74 LS 139. An den CTC wird im Micro-Professor ein Chip-Enable-Signal geliefert, wenn die Adreßleitungen A6 und A7 die Signal-Kombination Ø1 liefern.

Wir brauchen die grundlegenden Überlegungen zur Adressierung der Peripherie-Bausteine im Z 80-System nicht zu wiederholen. Ab der Seite H15 ist das ausführlich beschrieben worden. Bitte lesen Sie dort noch einmal nach. Hier geben wir gleich die fertige Tabelle für die Port-Adressierung der CTC-Kanäle an:

	A7	A6					CS1	CSØ			
Bit:	7	6	5	4	3	2	1	Ø	Adr.		Information
	Ø	Ø	X	X	X	X	Ø	Ø	4	Ø	Befehle für Kanal Ø
	Ø	Ø	X	X	X	X	Ø	1	4	1	Befehle für Kanal 1
	Ø	Ø	X	X	X	X	1	Ø	4	2	Befehle für Kanal 2
	Ø	Ø	X	X	X	X	1	1	4	3	Befehle für Kanal 3

Die Tabelle zeigt zunächst, daß die Adreß-Bits A7 und A6 den Bereich innerhalb der möglichen Port-Adressen bestimmen, in dem der CTC angesprochen wird. Die Adreß-Bits A1 und AØ sind für den angesprochenen CTC-Kanal zuständig.

Die Eintragungen X zeigen, daß auch die CTC-Adressen unvollständig decodiert sind. Was es mit dieser unvollständigen Decodierung auf sich hat, haben wir ebenfalls im Zusammenhang mit der Adressierung der PIO ausführlich erläutert.

Aufgabe H58.1

Überlegen Sie bitte, wieviele Port-Adressen durch die unvollständige Adressen-Decodierung des Z 80 CTC für andere Verwendungen unbrauchbar werden. Welche Adressen sind das?

Vergleichen Sie bitte die Aufgabe H22.1 und deren Lösung auf der Seite Ü2 mit der hier gestellten Aufgabe!

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü8.

Die in das Bild H57.1 blau eingetragenen Control-Leitungen für die Anschlüsse $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{M1}}$ und $\overline{\text{RESET}}$ entsprechen denen im Bild H19.1. Sie führen im Micro-Professor gemeinsam mit den zugehörigen Leitungen von der PIO zu den jeweils gleich bezeichneten CPU-Anschlüssen.

Mit dem Anschluß $\overline{INT}$ für Interrupt-Anforderungen werden wir uns gleich anschließend noch näher beschäftigen. Hier sei nur festgestellt, daß auch dieser Anschluß parallel mit dem der PIO an den INT-Anschluß der CPU geführt wird.

Interrupts über den CTC

Im Interrupt-Mode 2 der CPU, der für alle von Peripherie-Bausteinen eintreffenden Interrupt-Anforderungen zuständig ist (vgl. Seite H 37), wird der Interrupt immer über den CPU-Anschluß $\overline{INT}$ angefordert. Die CPU kann also nicht feststellen, welcher Peripherie-Baustein den Interrupt angefordert hat. Das will sie auch gar nicht wissen. Für die CPU ist allein entscheidend, welche Interrupt-Service-Routine sie aufrufen muß. Woher nimmt sie diese Information?

Wir haben Ihnen bereits gezeigt, daß die Anfangsadressen aller in einem Programm interessierenden Interrupt-Service-Routinen in einem Vektorfeld im Speicher abgelegt sind (Seite S 38). Der Inhalt des I-Registers in der CPU ist in einer Initialisierungs-Routine mit dem HOB der ersten Adresse des Vektorfelds geladen worden. Nach einer Interrupt-Anforderung muß der CPU also nur noch das LOB der Vektorfeld-Adresse mitgeteilt werden, bei der sie die Anfangsadresse der jeweils gewünschten Interrupt-Service-Routine abholen kann.

Bei der Beschreibung der Interrupt-Programmierung der PIO wurde erläutert, daß jedem PIO-Port per Programmierung ein eigenes Interrupt-Vektor-LOB zugeteilt wird. Wenn einer der PIO-Ports eine Interrupt-Anforderung an die CPU geschickt hat, dann wartet er, bis ihm über eine bestimmte Kombination der Signale $\overline{IORQ}$ und $\overline{M1}$ die Interrupt-Anforderung quittiert wird. Anschließend liefert er der CPU über den System-Datenbus sein ihm einprogrammiertes Interrupt-Vektor-LOB. Jetzt weiß die CPU, welche Interrupt-Service-Routine sie aufrufen muß.

Für Interrupt-Anforderungen durch den CTC gilt ganz genau das gleiche Prinzip. Auch hier hat jeder CTC-Kanal einen speziellen Interrupt-Vektor. Er wird dann auf den Datenbus des Systems geschaltet, wenn der Kanal über das *Channel Control Word* (vgl. Bild S 45.1) für Interrupt-Anforderungen programmiert wurde und der interne Abwärtszähler leergezählt worden ist.

Auf der Seite S 38 wurde begründet, daß es sehr sinnvoll ist, die Anfangsadressen der Interrupt-Service-Routinen für die PIO in einem Vektorfeld aufeinanderfolgend abzulegen. Grundsätzlich ist die Anlage eines Vektorfelds für die PIO allerdings nicht notwendig. Niemand kann Sie an dem Unfug hindern, den PIO-Ports bei der Programmierung weit auseinander weisende Interrupt-Vektoren zuzuordnen. Sie müssen nur dafür sorgen, daß die Bits Nr. Ø dieser Vektor-LOBs die Werte Ø haben.

Solchen Unfug zu machen, kommen Sie bei der Programmierung des CTC erst gar nicht in Versuchung. Dafür sorgt die bereits angedeutete, sehr einfache Programmierung des CTC, dem für alle vier Kanäle nur ein einziger Interrupt-Vektor übergeben werden muß. Es ist der für

den CTC-Kanal Ø. Wenn das einmal geschehen ist, dann verteilt der CTC intern die drei Interrupt-Vektoren für die restlichen drei Kanäle ganz selbständig. Dabei legt er automatisch ein Vektorfeld an: Jedem der Kanäle 1, 2 und 3 wird der Reihe nach ein Interrupt-Vektor zugeteilt, der sein Ziel gerade zwei Speicherzellen höher hat als der des vorhergehenden Kanals.

Im folgenden Software-Abschnitt zeigen wir Ihnen, was bei der Festlegung der CTC-Interrupt-Vektoren beachtet werden muß. Merken Sie sich vorerst nur, daß dem CTC bei der Programmierung der Kanäle nur ein einziges *Interrupt Vector Word* übermittelt werden muß.

Sehen Sie sich bitte zum Schluß noch im Bild H57.1 die Beschaltung der CTC-Anschlüsse IEI und IEO an! Sie erkennen, daß der CTC im Micro-Professor tatsächlich so eingebaut ist, wie es das Bild H41.1 zeigt. Interrupt-Anforderungen des CTC haben vor Interrupt-Anforderungen der PIO die höhere Priorität.

Das ist auch ganz einleuchtend. Wenn der CTC im *Timer Mode* nach dem Schema betrieben wird, wie es im Bild S41.1 gezeigt ist, dann hat er die Funktion einer Uhr, die ein Hauptprogramm in absolut festliegenden Intervallen für die Erledigung einer wichtigen Service-Routine unterbricht. Bei einer solchen Betriebsart ist es natürlich nicht zulässig, daß der Uhren-Takt nur deshalb nicht rechtzeitig wirksam werden kann, weil gerade irgendeine andere Peripherie über die PIO die Bearbeitung einer von ihr gelieferten Information wünscht.

Denkbar ist es auch, daß zwei Kanäle des CTC Uhren-Funktionen ausüben, die sich wegen unterschiedlicher Takt-Frequenzen zwangsläufig ab und zu ins Gehege kommen müssen. In diesem Fall muß der Programmierer entscheiden, welche der Uhren-Funktionen den Vorrang haben soll. Dieser Funktion kann innerhalb des CTC eine höhere Priorität zugeordnet werden. Möglich wird das durch die Tatsache, daß innerhalb des CTC bei den einzelnen Kanälen eine Interrupt-Hierarchie besteht: Die höchste Priorität haben die vom Kanal Ø angeforderten Interrupts. Den übrigen Kanälen sind mit steigender Kanal-Nummer geringere Prioritäten zugeordnet. Die niedrigste Priorität hat der Kanal 3.

Der Ein- und Ausgabebaustein 8255

Der Ein-/Ausgabebaustein mit der Bezeichnung 8255 ist ein universell einsetzbarer Mehrzweckbaustein für Mikroprozessorsysteme mit Mikroprozessoren der 80er-Familie, zu der auch der Z 80 gehört. Dieser Baustein ist einerseits auf der „Mikro-Professor"-Grundplatine zu finden, dort bedient er die Sieben-Segment-Anzeigen und die Tastatur, andererseits ist er als frei zugänglicher Baustein auch auf der Interface-Platine zu finden. Der Ein-Ausgabebaustein 8255 hat insgesamt 24 programmierbare Ein-Ausgänge. Diese können für verschiedene Zwecke entweder zu drei 8-Bit-Ports oder zu zwei 8-Bit-Ports und zwei 4-Bit-Ports zusammengefaßt werden. Wie bei den meisten Peripherie-Bausteinen wird auch beim 8255 Wert auf universelle Einsetzbarkeit gelegt. Die Funktion des Bausteins kann durch Software festgelegt werden, die in diesem Lehrbrief beschrieben wird.

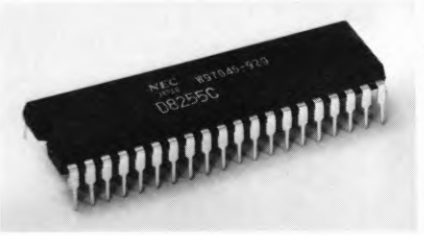


Bild H61.1
Der Ein-/Ausgabebaustein 8255 ist ein universeller Baustein mit insgesamt 24 Ein-Ausgabeleitungen, die in verschiedene Ports zusammengefaßt werden können.

Der durch die Software gewählte Betriebsmodus entscheidet über die Funktion dieses Bausteins, ganz ähnlich, wie Sie das bei der Programmierung der Z 80 PIO schon kennengelernt haben. Insgesamt stehen dem Anwender dieses Bausteins drei verschiedene Betriebsarten zur Auswahl, die in diesem Lehrgang einzeln besprochen werden, und für die Sie – soweit es die Möglichkeiten der Interface-Platine zulassen – Programme entwickeln und testen werden. Die prinzipielle Wirkungsweise der Betriebsarten – wir wollen diese auch wieder der internationalen Ausdrucksweise anpassen und Modus nennen – sei hier schon kurz angesprochen.

Der Modus 0

Die einfachste Betriebsart des 8255 ist der Modus 0. In diesem Modus arbeitet der Baustein als einfacher Ein-/Ausgabebaustein ohne Steuer- und Quittungssignale. Durch die IN- bzw. OUT-Befehle des Mikroprozessors können Daten in 4- oder 8-Bit-Struktur ausgesendet oder empfangen werden, wobei die Ports als Ein- oder Ausgabeports programmiert werden können.

Der Modus 1

Das Arbeiten im Modus 1 ist schon etwas komfortabler. Sechs Leitungen werden in diesem Modus für das sogenannte Handshaking benutzt, also für Steuer- und Quittungssignale im Betrieb mit anderen Peripheriegeräten. Die verbleibenden Leitungen können, wie auch schon im Modus 0 als Ein- und Ausgabeports programmiert werden.

Der Modus 2

Wenn in diesem Modus gearbeitet wird, kann ein Port mit 8 Bits als Ein- und Ausgabeport arbeiten. Ob Daten gesendet oder empfangen werden, entscheiden dann nur noch die entsprechenden Quittungssignale. Die verbleibenden Leitungen können als Ein- oder Ausgangsleitungen im Modus 0 oder 1 weiterarbeiten.

Der Modus 3

Der Modus 3 ist eigentlich eine Besonderheit des 8255 und weicht von den eben erwähnten Betriebsarten ein wenig ab. In diesem Modus können einzelne Bits eines Ports gesetzt oder rückgesetzt werden. Diese Funktion kann z. B. wichtig werden, wenn Steuer- oder Quittungssignale simuliert werden sollen.

Die interne Struktur des 8255

Um die Arbeitsweise des 8255 etwas genauer zu durchleuchten, ist es notwendig, zumindest in groben Zügen den internen Aufbau dieses interessanten Bausteins zu betrachten und die verschiedenen Portbezeichnungen und die Steuerleitungen zu untersuchen. Bild H62.1 zeigt den Aufbau des Bausteins in der Form eines Blockschaltbilds. Auf der rechten Seite des Bildes sehen Sie zunächst einmal die drei Ports mit den Bezeichnungen A, B und C. Die Ports A und B haben jeweils eine Datenbreite von 8 Bit bzw. einem Byte. Port C bildet eine Ausnahme, denn dieser Port kann entweder als 8-Bit-Port oder als zwei einzelne 4-Bit-Ports programmiert werden. Insgesamt vier Zwischenspeicher übernehmen den Datentransfer von der „Außenwelt" zum Mikroprozessorsystem und in die umgekehrte Richtung. Diese Zwischenspeicher sind notwendig, um Daten, die nur kurzfristig an den Ports anstehen, solange aufzubewahren, bis der Mikroprozessor vom Programmablauf her in der Lage ist, die Daten abzuholen und weiterzuverarbeiten. Alle Zwischenspeicher sind an den internen Datenbus angeschlossen.

Die drei Ports des 8255 werden intern von der Steuerung her in zwei Gruppen zusammengefaßt. Wir haben diese Gruppen der Übersichtlichkeit halber Gruppe X und Gruppe Y genannt. Die Funktion der Ports, also in welchem Modus sie arbeiten, wird durch zwei Steuerregister übernommen, die bei der Programmierung des Bausteins allerdings wie ein Register behandelt werden. Das soll im Moment aber nicht stören.

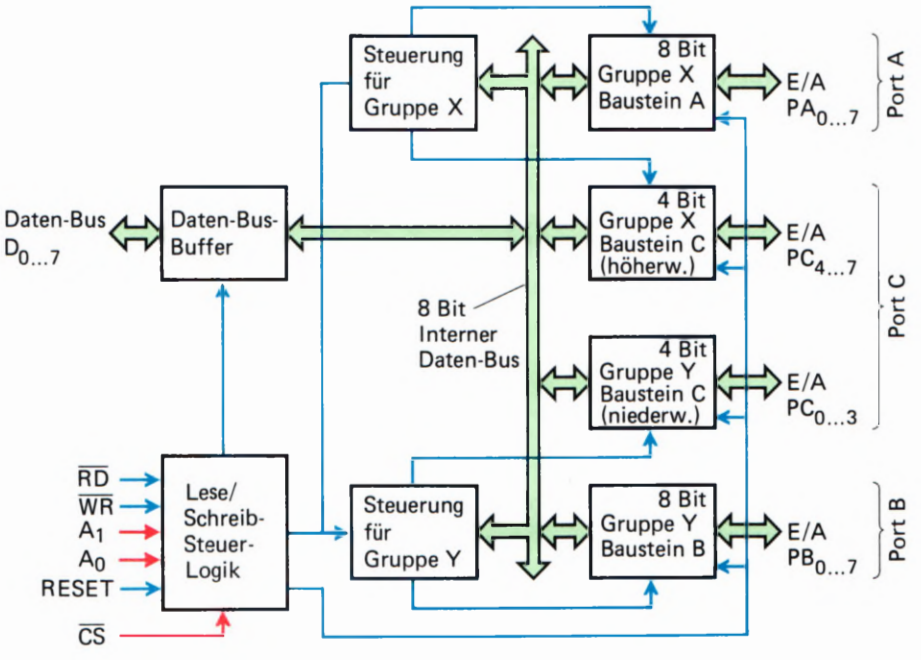


Bild H62.1
Das „Innenleben" des Bausteins 8255. Die wichtigsten Baugruppen sind die drei Ports, die Steuerregister und die Schreib-/Leselogik.

Links oben im Blockschaltbild sehen Sie den internen Datenbus-Buffer; darunterliegend die Steuerlogik dieses Bausteins, dessen Anschlüsse einer genaueren Betrachtung bedarf. Beginnen wir mit den Anschlüssen von unten nach oben.

Der Anschluß $\overline{CS}$ (C*hip*-S*elect* – Baustein-Anwahl) wird, wie bei Speicherbausteinen auch, dazu benutzt, den 8255 zu aktivieren. Ist an diesem Anschluß ein 1-Signal, so ist der Datenbus gegenüber dem Mikroprozessor-System hochohmig. Erst wenn ein Ø-Signal an diesem Eingang anliegt, ist der Baustein aktiv und sozusagen an das Bus-System des Mikroprozessors angeschlossen.

Der Anschluß RESET (Rücksetzen) ist normalerweise mit dem RESET-Anschluß des Mikroprozessor-Systems gekoppelt. Dieser hat auch die gleiche Funktion, wie der RESET-Anschluß des Mikroprozesssors: Er programmiert den Baustein in ganz bestimmter Weise. Beim 8255 hat dieser Anschluß die Funktion, die Ports des Bausteins auf Eingabe zu programmieren, und zwar im Modus 0. Alle Daten der internen Zwischenspeicher, auch das Steuerwortregister, werden gelöscht. Der RESET-Anschluß ist aktiv, wenn ein 1-Signal an diesem Eingang anliegt.

Die Anschlüsse AØ und A1. Egal in welchem Modus der 8255 betrieben wird, sobald Daten vom Mikroprozessor zu einem Port des 8255 gelangen sollen, oder umgekehrt, so muß dem Baustein „gesagt werden", welcher Port gemeint wird. Sind beide Anschlüsse mit Ø-Signal belegt, dann ist der Port A gemeint. Ist an AØ ein Ø-Signal und an A1 ein 1-Signal, so ist der Port B angesprochen. Der Port C ist schließlich angesprochen, wenn an AØ ein 1-Signal und an A1 ein Ø-Signal anliegt. Die vierte Möglichkeit, daß an beiden Anschlüssen ein 1-Signal anliegt, aktiviert das Steuerwortregister. Für den 8255 ist das ein Zeichen, daß in diesem Fall auf dem Datenbus ein Steuerwort anliegt, das in das Steuerwortregister eingeschrieben werden soll. Die beiden Anschlüsse AØ und A1 sind an den Adreßbus des Mikroprozessors angeschlossen. Die verschiedenen Möglichkeiten sind in folgender Tabelle noch einmal zusammengefaßt.

AØ	A1	Aktiver Port
Ø	Ø	Port A ist aktiviert
1	Ø	Port B ist aktiviert
Ø	1	Port C ist aktiviert
1	1	Steuerwortregister ist aktiviert

Der Anschluß $\overline{RD}$ (R*ead* – Lesen) bekommt immer dann ein Signal vom Mikroprozessor, wenn Daten vom 8255 zum Mikroprozessor gelangen sollen. Dieses Signal ist aktiviert, wenn ein Ø-Signal an diesem Anschluß anliegt. Darauf deutet auch der Querstrich über der Anschlußbezeichnung im Blockschaltbild. Der Mikroprozessor behandelt in diesem Fall den Baustein ähnlich, als ob er Daten von einem Speicherbaustein liest.

Der Anschluß $\overline{WR}$ (**WR***ite* – Schreiben) hat im Grunde die umgekehrte Funktion wie der Anschluß $\overline{RD}$. Dem 8255 wird durch ein Ø-Signal an diesem Eingang vom Mikroprozessor bekannt gegeben, daß Daten von diesem zu erwarten sind, die in den 8255 gelangen sollen.

Durch die beiden Anschlüsse $\overline{RD}$ und $\overline{WR}$ hat der Anwender die Möglichkeit, diesen Baustein auch an Systeme anzuschließen, in denen Mikroprozessoren eingesetzt werden, die nicht über einen IN- und OUT-Befehl verfügen. Diese Mikroprozessoren behandeln die Ein-/Ausgabebausteine wie ganz „normale" Speicherbausteine.

Die Sockelbeschaltung des 8255

Das Bild H64.1 zeigt die Anschlußbelegung des Ein-/Ausgabebausteins 8255. Sie finden die eben in Kurzform besprochenen Steueranschlüsse, die Anschlüsse für die Spannungsversorgung – der 8255 wird mit +5 V und Masse versorgt – und die Anschlüsse der drei Ports, die mit A, B und C bezeichnet sind. Weitere 8 Anschlüsse führen zum Datenbus des jeweiligen Mikroprozessor-Systems, an das der 8255 angeschlossen ist.

Zu beachten ist, daß die Ausgänge des Bausteins, also die Ports, nur mit 1 mA belastet werden dürfen. Will man bestimmte Baugruppen oder Geräte direkt durch den 8255 ansteuern, ist zu beachten, welchen Strom diese „ziehen", weil ansonsten der Baustein kaputt gehen kann und sich für immer „verabschiedet". Eventuell sollte man die Ausgänge des Bausteins puffern. Der Ausgangspegel der Ports ist ebenfalls zu beachten, wenn Peripherie direkt betrieben wird, er ist 1,5 V. Das reicht als 1-Pegel für die meisten TTL- oder CMOS-Bausteine aus. Bei manchen „Exoten" sollten Sie sich allerdings vergewissern, daß der Pegel auch wirklich ausreicht. Ansonsten ist der 8255 ein äußerst robuster Baustein, der – laut Datenblatt – zumindest für drei Sekunden auch einmal einen Kurzschluß an seinen Port-Ausgängen verträgt. Eine Eigenschaft, die nicht jeder Baustein hat und die von manchem Entwickler durch ein hörbares Aufatmen geschätzt wird, wenn, z.B. durch eine Meßstrippe, kurzfristig ein Ausgang kurzgeschlossen ist.

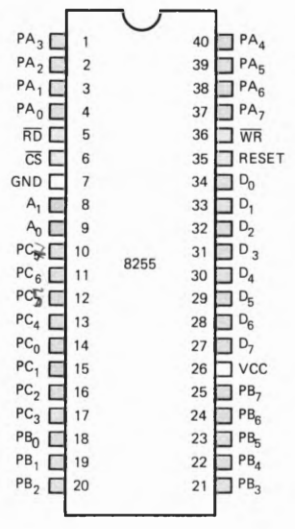


Bild H64.1
Die Sockelbeschaltung des 8255. Der Daten-, Adreß- und Steuerbus sind in verschiedenen Farben dargestellt.

Bezeichnungen

D_0 - D_7	Datenbus
PA_0 - PA_7	Port A
PB_0 - PB_7	Port B
PC_0 - PC_7	Port C
V_{CC}	+5 V
GND	Masse
$\overline{RD}$	READ (Lesen)
$\overline{WR}$	WRITE (Schreiben)
$\overline{CS}$	CHIP SELECT
A_0, A_1	Adresse für Ports

Der Anschluß des 8255 auf der Peripherie-Leiterplatte

Der 8255 ist auf der Peripherie-Leiterplatte fest verdrahtet. Für den Betrieb des Bausteins ist nur die Verbindung der Leiterplatte mit dem „Micro-Professor" notwendig. Diese Verbindung stellen Sie bitte durch das 40-polige Flachbandkabel her, wie es schon in Lehrbrief 1 (Seite H 10) des Lehrgangs beschrieben wurde. Wir verzichten an dieser Stelle auf eine detaillierte Beschreibung und bitten Sie, bei Bedarf auf der Seite H 10 nachzulesen.

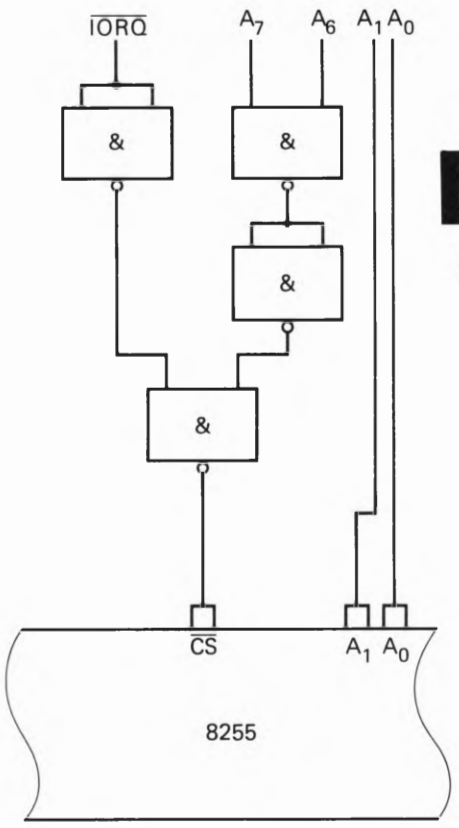


Bild H 65.1
Durch die kleine Logik-Anordnung werden vom Adreßbus die Leitungen A0, A1 und CS bedient.

Der Adreßbus

Daß zum Programmieren von Mikroprozessoren und speziell der Peripheriebausteine auch Kenntnisse der Hardware notwendig sind, wird Ihnen bei der Programmierung des 8255 besonders auffallen. Der 8255 ist durch das Flachbandkabel fest mit dem Datenbus des Z 80 verbunden. Vier Leitungen des Adreßbus' sind ebenfalls fest mit dem 8255 verbunden. Zwei Leitungen, und zwar A0 und A1, wählen den gewünschten Port aus. Zwei weitere Adreßleitungen, A6 und A7, bedienen zusammen mit einer kleinen Logik-Anordnung und dem IORQ-Signal des Z 80 die Bausteinanwahl CS. Wäre der 8255 nur an die Adreßleitungen angeschlossen, würde der Mikroprozessor den Ein-/Ausgabebaustein wie einen „normalen" Speicher behandeln. Durch das **IORQ-Signal** (*In Out ReQuest* – Ein-Ausgabe-Quittungssignal), das der Z 80 ausgibt, kann man unterscheiden, ob der Mikroprozessor mit einem Speicher oder mit einem Peripherie-Baustein korrespondieren will. Hat das IORQ-Signal einen 0-Pegel, so ist ein Peripherie-Baustein gemeint. Umgekehrt, wenn ein 1-Pegel an diesem Ausgang des Mikroprozessors liegt, ist ein Speicherbaustein gemeint. Damit kann man erreichen, daß durch eine geeignete Codierung insgesamt 255 Peripheriebausteine getrennt angesprochen werden können. Das IORQ-Signal haben wir uns auf der Peripherie-Leiterplatte zunutze gemacht. Bild H 5.1 zeigt die Logik-Anordnung, mit der das CS-Signal bedient wird. Daraus können Sie erkennen, daß der 8255 nur dann angesprochen ist, wenn die Adreßleitungen A6 und A7 einen 1-Pegel aufweisen und das IORQ-Signal einen 0-Pegel hat.

Die Adreßcodierung

An dieser Stelle müssen wir einen Schritt zurückgehen, und zwar zur Software des Mikroprozessors. Wie Sie nun schon wissen, werden Ein-/Ausgabebausteine durch den Befehl IN und den Befehl OUT angesprochen. Beide Befehle sind Zweibyte-Befehle. Das erste Byte ist das Befehlsbyte, das zweite Byte ist das Adreßbyte. Daß die Adresse bei der Ein-/Ausgabe von Daten nur ein Byte breit ist, ist eine Besonderheit, die uns hier aber nicht stören soll. Wichtiger ist es für den Praktiker, daß er weiß, wie der Ein-/Ausgabebaustein „hardwaremäßig" an das Mikroprozessorsystem angeschlossen ist, damit er festlegen kann, wie das Adreßbyte aussehen muß, um den Baustein anzusprechen. Wir haben bereits erklärt, daß der 8255 auf der Peripherie-Leiterplatte durch vier Bit des Adreßbus' angesprochen wird. Dadurch ergeben sich – bedingt durch die Hardware – folgende Möglichkeiten für das Adreßbyte des IN- und OUT-Befehls: **C0, C1, C2, C3.**

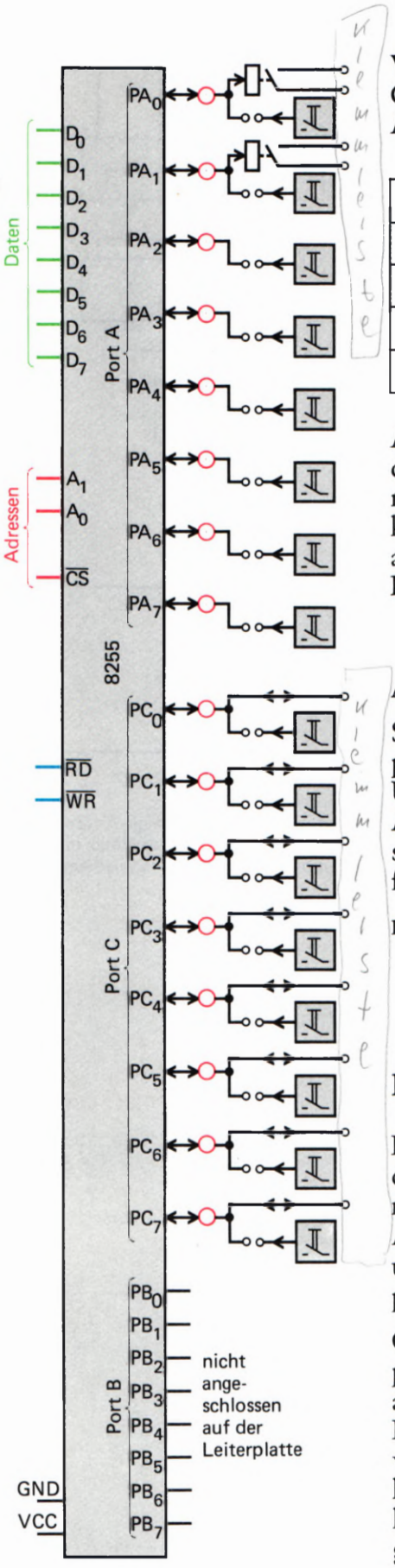


Bild H66.1
Die Ports des 8255 können entweder als Eingänge oder als Ausgänge geschaltet werden.

Wie kommen wir auf diese Hexadezimalcodierungen? Nun, das ist im Grunde ganz einfach. Man muß sich nur einmal die Binärstruktur der Adresse ansehen.

A7	A6	A5	A4	A3	A2	A1	A0	Hex	Bedeutung
1	1	X	X	X	X	0	0	C0	Port A
1	1	X	X	X	X	0	1	C1	Port B
1	1	X	X	X	X	1	0	C2	Port C
1	1	X	X	X	X	1	1	C3	Steuerwort

Aus der Tabelle können Sie genau verfolgen, wie aus der Binärstruktur der 8 Adreßleitungen die Adressen für den 8255 gebildet werden. Die mit „X" gekennzeichneten Adreßleitungen sind nicht benutzt und können 0- oder 1-Pegel annehmen, ohne daß sich bei der Bausteinanwahl etwas ändert. Wir haben schon bei der Besprechung der Z 80 PIO eine ähnliche unvollständige Codierung vorgefunden.

Aufgabe H66.1

Stellen Sie sich vor, daß ein 8255 so mit dem Adreßbus eines Mikroprozessor-Systems verbunden ist, wie es Bild H65.1 zeigt. Einziger Unterschied soll sein, daß die Adreßleitungen A0 und A1 an den Adreßleitungen A4 und A5 des Mikroprozessor-Systems angeschlossen sind. Welche Hexadezimalcodierungen ergeben sich in diesem Fall für Port A, B und C sowie für das Steuerwort?

Die Lösung finden Sie auf der Seite Ü11.

Die Ausgänge des 8255

Insgesamt stehen dem Anwender des 8255 drei Ports zur Verfügung, das wissen Sie nun bereits. Um die Hardware zu diesem Lehrgang nicht unnötig zu verteuern, haben wir uns darauf beschränkt, die Ports A und C auf der Peripherie-Leiterplatte mit den Leuchtdioden, Relais und Tastern zu verbinden. Der Port B ist auf der Peripherie-Leiterplatte leider nicht zugänglich.

Grundsätzlich sind die Ports A und C mit den Leuchtdioden der Peripherie-Leiterplatte verdrahtet. Die untere Leuchtdiodenreihe (siehe auch Bild H11.1) ist mit dem Port A des 8255 verbunden; die obere Leuchtdiodenreihe ist mit dem Port C des 8255 verbunden. Genauso, wie Sie es bei der Besprechung der Z 80 PIO schon kennengelernt haben, können Sie auch die Ports des 8255 durch die steckbaren Brükken als Eingänge benutzen. Es ist also alles sozusagen „beim alten" geblieben. Was die Peripherie-Leiterplatte allerdings nicht zuläßt ist, daß Sie den 8255 und die Z 80 PIO gleichzeitig benutzen. In diesem Falle würden an den Leuchtdioden Pegel angezeigt werden, die keine eindeutige Zuordnung zum jeweiligen Peripherie-Baustein mehr zulassen.

Die Eingangssignale werden wahlweise durch die beiden Reihen von Tastern oder die beiden Gruppen von Schiebeschaltern erzeugt. Die obere Schalter- bzw. Tasterreihe kann mit dem Port C des 8255 verbunden werden. Die untere Reihe wird über die steckbaren Brücken mit dem Port A des 8255 verbunden.

Die Belegung der Anschlußklemmleiste

Die Anschlußklemmleiste auf der Peripherie-Leiterplatte wird hauptsächlich vom Port C des 8255 bedient, damit Sie die Möglichkeit haben periphere Geräte, die Quittungssignale benötigen, an die Leiterplatte anzuschließen. Die mit den Ziffern 0 bis 7 gekennzeichneten Anschlüsse werden vom Port C bedient und sind sozusagen immer mit dem Baustein verbunden.

Die mit Rel. 1 und Rel. 2 bezeichneten Anschlüsse sind die Relaisausgänge. Das Relais 1 kann durch Bit Ø des Port A angesteuert werden; das Relais 2 kann durch Bit 1 des Port A angesteuert werden. Die Relais-Ausgänge werden dann wichtig, wenn Sie potentialfrei schalten wollen und keine galvanische Kopplung gebrauchen können.

Sie sind ja von den Lehrbriefen 1 bis 3 her die Benutzung der Peripherie-Leiterplatte schon sehr gut gewöhnt, so daß es einer weiteren detaillierten Erklärung der Funktion nicht mehr bedarf. Wenden wir uns also wieder speziell dem 8255 zu. Wir beginnen gleich mit der Programmierung in der Betriebsart 0 des Bausteins.

Die Betriebsart 0

Von den verschiedenen Betriebsarten des Ein-/Ausgabebausteins 8255 ist die Betriebsart 0 die einfachste. Jeder der drei Ports kann als Eingabe- oder als Ausgabeport arbeiten. Dabei weist der Port C eine Besonderheit auf: Er kann durch ein entsprechendes Steuerwort in zwei 4-Bit-Ports gesplittet werden, die wiederum als Ein- und Ausgabeports arbeiten können. Insgesamt sind damit 16 verschiedene Kombinationsmöglichkeiten in der Betriebsart 0 gegeben.

Bild H67.1 zeigt die möglichen Kombinationen der Ports in der Betriebsart 0. Ist ein Port auf Ausgabe programmiert worden, so stehen die vom Mikroprozessor durch einen OUT-Befehl gelieferten Daten solange an diesem Port an, bis neue Daten geliefert werden. Die Zwischenspeicherung ist auch notwendig, weil man den Zeitpunkt, zu dem die anstehenden Daten abgeholt werden, nicht genau bestimmen kann. In der Betriebsart 0 werden vom 8255 keine Quittungssignale geliefert oder in Empfang genommen. Das ist Ihnen bei der Betrachtung von Bild H67.1 bestimmt schon aufgefallen.

In umgekehrter Richtung, wenn Daten von der Peripherie über den 8255 zur CPU gelangen sollen, wird es etwas schwieriger, weil die von außen kommenden Daten im Baustein nicht zwischengespeichert werden. Wenn Sie Daten in diesem Modus einlesen wollen, muß die Software des Mikroprozessors diesen veranlassen, solange in einer Warte-

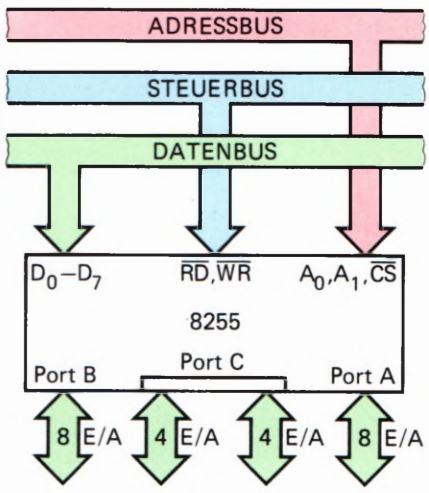


Bild H67.1
Der Anschluß des 8255 an das Bussystem des Mikroprozessors. Insgesamt stehen dem Anwender des 8255 drei Ports zur Verfügung, die unterschiedliche Funktionen erfüllen können.

schleife zu verweilen, bis die einzulesenden Daten am entsprechenden Port des 8255 anstehen. Sie werden dies gleich in einem Beispielprogramm praktisch erfahren. Zuvor werden wir erst einmal untersuchen, wie der 8255 in den Modus 0 gebracht wird.

Einschreiben des Steuerworts in den 8255

Ehe überhaupt mit dem Baustein 8255 gearbeitet werden kann, muß er für die jeweiligen Wünsche des Anwenders programmiert werden, das ist fast so wie beim Telefonieren. Zuerst muß die richtige Telefonnummer gewählt werden, erst dann wird die richtige Verbindung mit dem gewünschten Teilnehmer hergestellt, und der Datenfluß kann beginnen. Der 8255 wird durch ein Steuerwort programmiert.

Das Steuerwort gelangt durch einen OUT-Befehl in das Steuerwortregister. Die Adresse für den OUT-Befehl muß so gewählt werden, daß die Adreßleitungen, an die der 8255 angeschlossen ist, beide eine logische 1 aufweisen. In unserem Fall ist die Adresse in sedezimaler Schreibweise C3H. Bitte vergleichen Sie hierzu das Bild H65.1. Die Adreßleitungen A0, A1, A6 und A7 müssen auf logisch 1 liegen, denn nur dann wird der Baustein angesprochen und das Steuerwort in das Steuerregister eingeschrieben.

Der Aufbau des Steuerworts

Jedes Bit des Steuerworts hat eine bestimmte Bedeutung. Bild H69.1 zeigt schematisch die Bedeutung der einzelnen Bits.

D0 entscheidet über die Datenflußrichtung des Ports C, und zwar über die niederwertigen 4 Bits. Eine logische 1 bedeutet, daß dieser Teil des Ports C auf Daten-Eingabe programmiert wird. Eine logische 0 bedeutet hingegen, daß dieser Teil des Ports C auf Datenausgabe programmiert ist.

D1 entscheidet über die Datenflußrichtung des Ports B, und zwar über die gesamten 8 Bits. Auch für diesen Port gilt: Eine logische 1 bedeutet Eingabe; eine logische 0 bedeutet Ausgabe.

D2 entscheidet über die Betriebsart des Ports B und die Betriebsart der niederwertigen Hälfte des Ports C. Eine logische 0 an D2 bedeutet Betriebsart 0 und konsequenterweise eine logische 1 die Betriebsart 1. Hieraus geht schon hervor, daß der 8255 die Betriebsart 2 für die Ports B und C nicht zuläßt. Wohl aber eine Mischung verschiedener Betriebsarten unterschiedlicher Ports. Auf diese Mischung gehen wir aber erst zum Schluß des Lehrgangs ein.

D3 entscheidet über die Datenflußrichtung der höherwertigen 4 Bits des Ports C. Wie schon erwartet, gelten auch hier die gleichen Regeln wie oben besprochen: 1 = Eingabe, 0 = Ausgabe.

D4 schließlich entscheidet, ob der Port A auf Dateneingabe oder auf Datenausgabe programmiert ist. Auch bei diesem Port gelten die gleichen Regeln wie unter D3 schon besprochen.

Bedeutung des Steuerworts

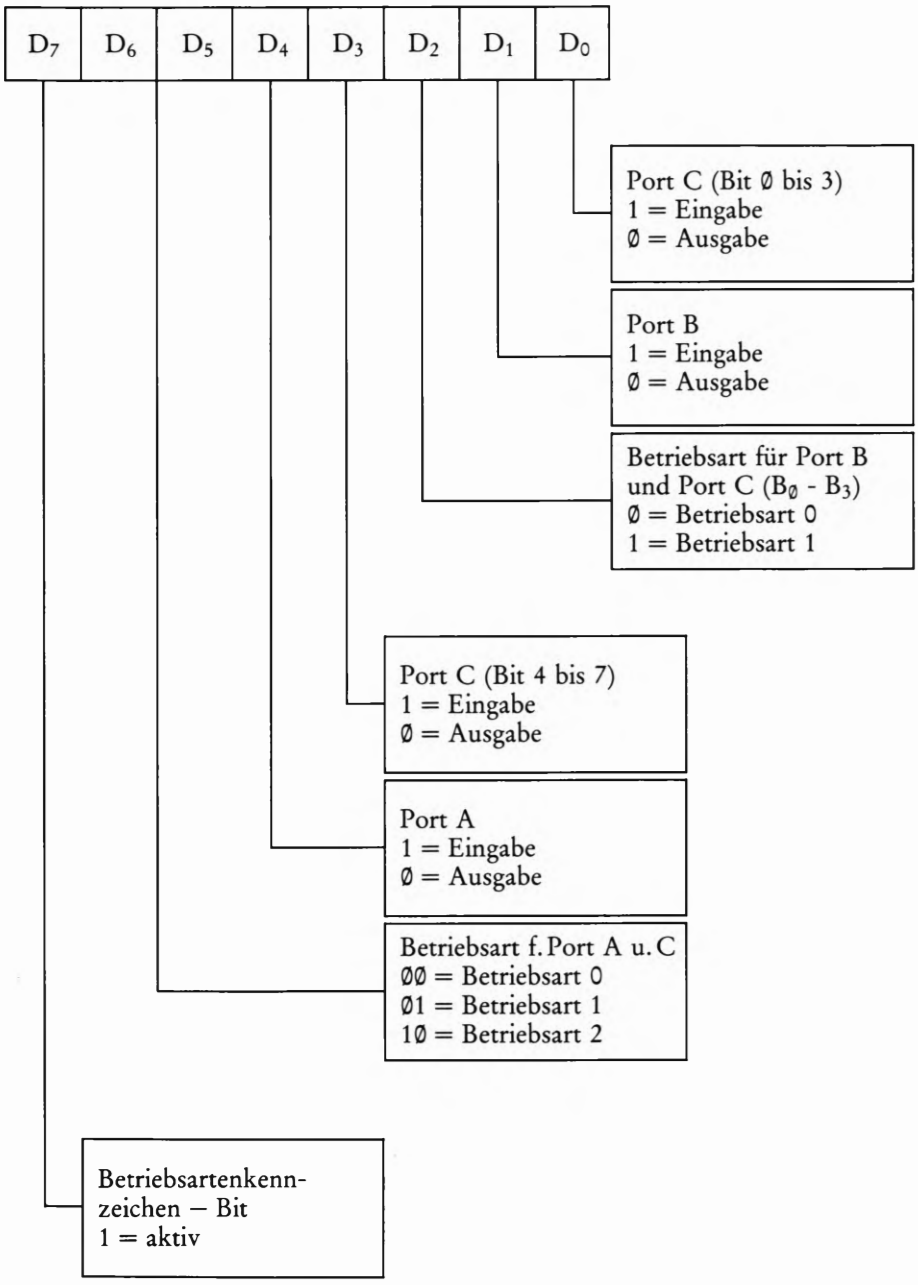


Bild H69.1
Jedes Bit des Steuerworts hat eine ganz bestimmte Bedeutung. Schon in der Betriebsart 0 kann man 16 verschiedene Steuerwörter wählen.

D5/D6 Hierdurch entscheidet sich, in welcher Betriebsart der Port A und die höherwertigen 4 Bits des Ports C betrieben werden. Für die Auswahl der Betriebsarten sind hierbei zwei Bits erforderlich, weil drei Betriebsarten gewählt werden können.

D7 hat eine besondere Bedeutung, auf die wir später eingehen werden. Vorerst soll uns die Feststellung genügen, daß an dieser Stelle des Steuerworts eine 1 stehen muß.

Nun haben wir also das notwendige Rüstzeug, um die Bedeutung des Steuerworts zu verstehen. Um Anfangsschwierigkeiten zu umgehen, haben wir einmal die verschiedenen Möglichkeiten – nur in der Betriebsart 0 – in einer Tabelle zusammengestellt. Sie sehen, daß es allein in dieser Betriebsart schon insgesamt 16 verschiedene Kombinationsmöglichkeiten der Ports gibt.

Tabelle H70.1

D_7	D_6	D_5	D_4	D_3	D_2	D_1	$D_\emptyset$	Sedezimal	PORT A	PORT B	PORT C C_7-C_4	PORT C C_3-$C_\emptyset$
1	Ø	Ø	Ø	Ø	Ø	Ø	Ø	80	Ausgang	Ausgang	Ausgang	Ausgang
1	Ø	Ø	Ø	Ø	Ø	Ø	1	81	Ausgang	Ausgang	Ausgang	Eingang
1	Ø	Ø	Ø	Ø	Ø	1	Ø	82	Ausgang	Eingang	Ausgang	Ausgang
1	Ø	Ø	Ø	Ø	Ø	1	1	83	Ausgang	Eingang	Ausgang	Eingang
1	Ø	Ø	Ø	1	Ø	Ø	Ø	88	Ausgang	Ausgang	Eingang	Ausgang
1	Ø	Ø	Ø	1	Ø	Ø	1	89	Ausgang	Ausgang	Eingang	Eingang
1	Ø	Ø	Ø	1	Ø	1	Ø	8A	Ausgang	Eingang	Eingang	Ausgang
1	Ø	Ø	Ø	1	Ø	1	1	8B	Ausgang	Eingang	Eingang	Eingang
1	Ø	Ø	1	Ø	Ø	Ø	Ø	90	Eingang	Ausgang	Ausgang	Ausgang
1	Ø	Ø	1	Ø	Ø	Ø	1	91	Eingang	Ausgang	Ausgang	Eingang
1	Ø	Ø	1	Ø	Ø	1	Ø	92	Eingang	Eingang	Ausgang	Ausgang
1	Ø	Ø	1	Ø	Ø	1	1	93	Eingang	Eingang	Ausgang	Eingang
1	Ø	Ø	1	1	Ø	Ø	Ø	98	Eingang	Ausgang	Eingang	Ausgang
1	Ø	Ø	1	1	Ø	Ø	1	99	Eingang	Ausgang	Eingang	Eingang
1	Ø	Ø	1	1	Ø	1	Ø	9A	Eingang	Eingang	Eingang	Ausgang
1	Ø	Ø	1	1	Ø	1	1	9B	Eingang	Eingang	Eingang	Eingang

Aufgabe H70.1

Bitte tragen Sie in die obenstehende Tabelle selbst einmal die hexadezimalen Zahlenwerte für die verschiedenen Steuerwörter ein.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü11.

Der 8255 auf der Micro-Professor-Platine

Eine „echte" Anwendung findet der 8255 auf der Micro-Professor-Platine. Dort werden die 7-Segmentanzeigen durch den Port B und den Port C gesteuert. In einer sehr raschen Folge werden alle Segmente der sechs 7-Segmentanzeigen kurz angesteuert, so daß der Eindruck entsteht, daß alle Anzeigen gleichzeitig in Betrieb sind. Multiplexbetrieb nennt man diese Vorgangsweise. Sie werden sich vielleicht fragen, wieso man diese auf den ersten Blick seltsame Vorgangsweise wählt.

Nun, das ist ganz einfach zu erklären: Der Multiplexbetrieb erspart eine ganze Menge Hardware. Würde man tatsächlich alle 7-Segmentanzeigen gleichzeitig ansteuern, so wäre der Stromverbrauch auf dem Prozessorsystem wesentlich größer. Die Folge wäre, daß das Netzgerät wesentlich teurer werden würde. Außerdem müßte jedes Segment jeder einzelnen 7-Segmentanzeige eine eigene Treiberstufe bekommen. Auch dies hätte einen wesentlich größeren Hardware-Aufwand zur Folge – keine Frage, das System würde erheblich teurer werden. Der Multiplexbetrieb – Software gesteuert – erspart hier also manchen Hardware-Aufwand. Bild H72.1 zeigt einen Auszug aus dem Schaltplan des Micro-Professor-Entwicklungssystems.

Gerade anhand dieser Hardware-Anordnung wird ganz deutlich, daß Hardware und Software bei der Entwicklung irgendeines Mikroprozessorsystems nicht ohne weiteres voneinander zu trennen sind. Es kann in diesem Fachgebiet, wenn es ernsthaft betrieben wird, keinen Hardware-Entwickler geben, der keine Ahnung von der Software eines Mikroprozessorsystems hat. Genauso ist der umgekehrte Fall fast undenkbar.

Noch gravierender ist die Situation des Service-Technikers. Wie soll er einen Fehler in einem System lokalisieren, wenn er zum Beispiel nur über die Hardware Bescheid weiß? Nehmen Sie nur einmal den Fall an, daß die Anzeige eines Systems nicht funktioniert. Dies kann sicherlich im einfachsten Fall eine defekte 7-Segmentanzeige oder ein defekter Treiberbaustein sein. Andererseits ist es aber auch durchaus denkbar, daß ein Fehler im Programm vorliegt. Aber nun zurück zum 8255 auf der Micro-Professor-Platine.

Port A des Bausteins übernimmt die Abfrage der Tastatur. Die sogenannte Matrix-Anordnung der Tasten wird kontinuierlich auf Tastendruck abgefragt. Ist eine Taste gedrückt worden, sorgt die entsprechende Software dafür, daß die Bedeutung der Taste erkannt und dekodiert wird. Auch hier sehen Sie deutlich das Zusammenspiel von Hardware und Software eines Mikroprozessorsystems.

Als dritte Funktion übernimmt der 8255 die Bedienung des eventuell an das System angeschlossenen Kassettenrecorders. Über das Bit C7 des Ports C werden Daten seriell an den Kassettenrecorder abgegeben und gleichzeitig über einen Transistor auch auf einen kleinen Lautsprecher übertragen. In umgekehrter Richtung arbeitet das Bit A7 des Ports A. Über diese Leitung gelangen Daten vom Kopfhörerausgang des Kassettenrecorders in die RAMs des Mikroprozessorsystems. Sie sehen, der 8255 wird auf der Micro-Professor-Platine ganz universell eingesetzt.

H

72

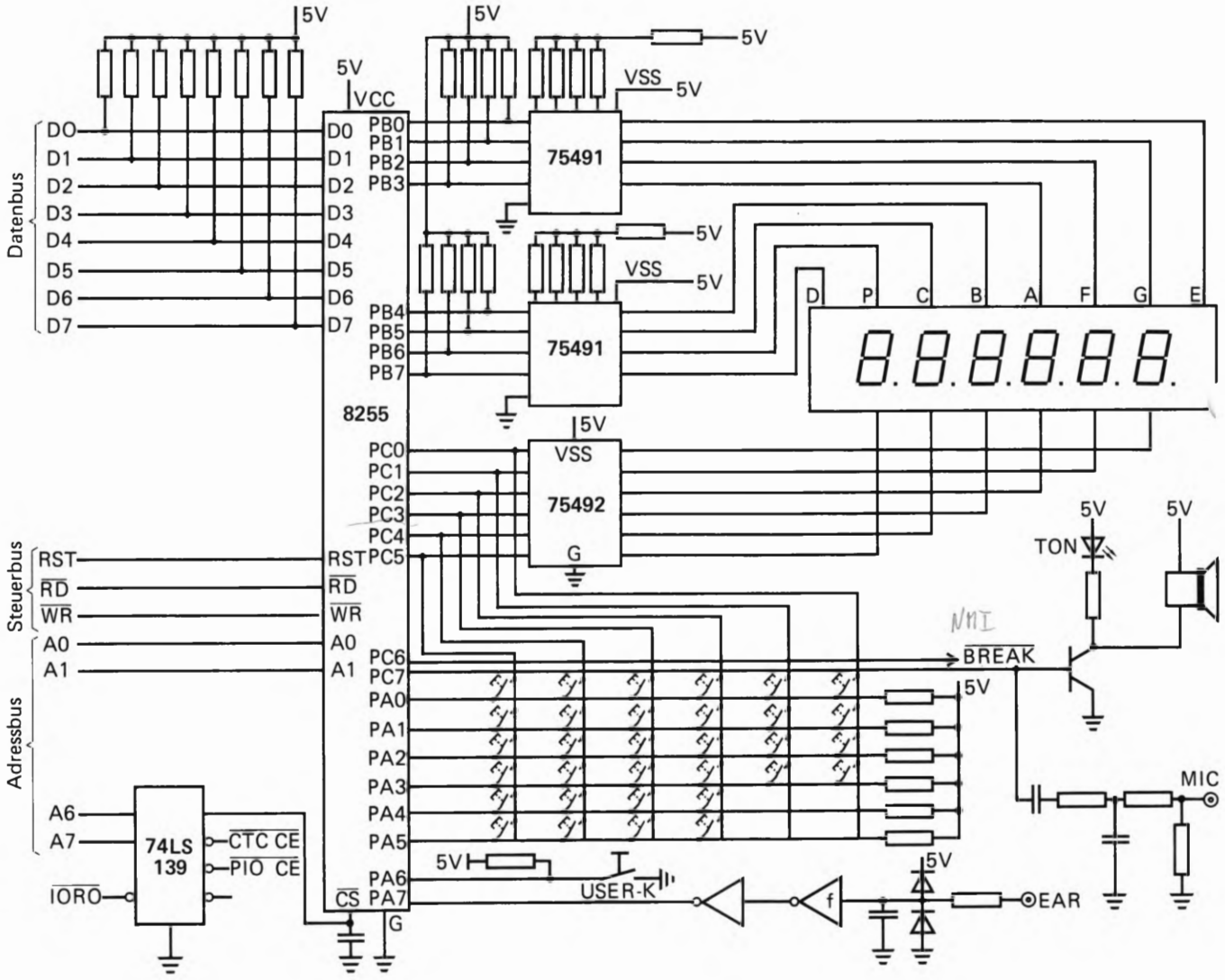
Anschluß des 8255 auf der Micro-Professor-Platine

Ehe man mit dem Baustein 8255 arbeiten kann, muß man wissen, wie er an den Adreß-, Daten- und Steuerbus des Mikroprozessorsystems angeschlossen ist. Daraus ergeben sich dann die entsprechenden Steuerworte und Adressen zur Aktivierung der einzelnen Ports des Bausteins. Das kennen Sie ja bereits aus der Besprechung des 8255 auf der Peripherie-Platine (siehe Seite H65).

Wenn Sie das in Bild H72.1 dargestellte Schaltbild betrachten, sehen Sie, daß der 8255 zunächst mit seinen Datenleitungen parallel zum Datenbus des Systems angeschlossen ist. Jede Datenleitung ist über einen 10-K-Ohm Widerstand mit der 5 V Spannung des Systems verbunden (auf 5 V hochgezogen, sagt der Fachmann). Diese Maßnahme entlastet den Datenbus des Systems und soll uns im Moment nicht weiter interessieren.

Die Anschlüsse Read ($\overline{RD}$) und Write ($\overline{WR}$) sind mit dem Z 80 Steuerbus verbunden. Über diese Leitungen wird dem 8255 mitgeteilt, welche Richtung die Daten über den Datenbus nehmen sollen. Mit anderen Worten: Ist Read aktiv, heißt dies, daß Daten vom 8255 zum Z 80 gelangen sollen; ist Write aktiv, sollen Daten vom Z 80 zum 8255 gelangen.

Bild H72.1
Der 8255 auf der Micro-Professor-Platine steuert die 7-Segment-Anzeige, die Tastatur und übernimmt den Datenverkehr mit dem Kassettenrekorder.



Die Adreßleitungen AØ und A1 sind mit dem System Adreßbus verbunden – AØ mit dem Bit Ø und A1 mit dem Bit 1 des Adreßbus. Zwei weitere Adreßleitungen sind für den 8255 noch von großer Bedeutung. Es sind dies die Adreßleitungen A6 und A7. Über diese beiden Leitungen wird durch den Multiplexer 74139 entschieden, ob die Z 80 PIO, der CTC-Baustein oder der 8255 angesprochen wird. Der 8255 wird dann aktiviert, wenn beide Adreßleitungen eine logische Null aufweisen. Dann nämlich bekommt der $\overline{CS}$-Eingang des 8255 ein Aktivierungssignal. Mit diesem Wissen ausgerüstet, können Sie die Adressen der Ports und die Adressen des Steuerwortregisters leicht selbst festlegen. Er ergeben sich folgende Adressen für die OUT- und IN-Befehle:

A7	A6	A5	A4	A3	A2	A1	AØ	Hex	Bedeutung
Ø	Ø	X	X	X	X	Ø	Ø	ØØ	Port A
Ø	Ø	X	X	X	X	Ø	1	Ø1	Port B
Ø	Ø	X	X	X	X	1	Ø	Ø2	Port C
Ø	Ø	X	X	X	X	1	1	Ø3	Steuerwortreg.

Die mit einem „X“ gekennzeichneten Bits sind für die Codierung bedeutungslos. Sie können mit einer Ø oder mit einer 1 belegt werden. Wir sind bei der Bildung der Sedezimalcodierung davon ausgegangen, daß diese Bits mit jeweils einer Ø belegt werden.

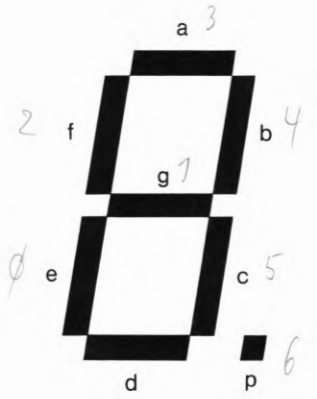
Damit sind nun alle „Fakten“ der Eingangsseite des 8255 festgelegt. Welche Funktion die einzelnen Ports des Bausteins haben, soll gleich besprochen werden.

Die Funktion des Ports B

Durch den Port B des 8255 auf der Micro-Professor-Platine werden die einzelnen Segmente der Anzeigen angesteuert. Jedem einzelnen Bit des Ports ist ein Segment einer 7-Segmentanzeige zugeordnet. Die Entwickler des Micro-Professor-Systems haben folgende Zuordnung gewählt:

a = Bit Nr. 3
b = Bit Nr. 4
c = Bit Nr. 5
d = Bit Nr. 7

e = Bit Nr. Ø
f = Bit Nr. 2
g = Bit Nr. 1
Punkt p = Bit Nr. 6



Wenn Sie sich die Verteilung der Segmente auf die einzelnen Bits des Ports B ansehen, werden Sie feststellen, daß auf den ersten Blick eine gewisse Unregelmäßigkeit herrscht. Die Verteilung hat aber ihren Sinn darin, daß das Umcodierungsprogramm einfacher wird. Das Umcodierungsprogramm ist deshalb unumgänglich, weil die Zahl oder der Buchstabe, der angezeigt werden soll, nicht dem notwendigen Bitmuster für die einzelnen Segmente entspricht. Besser als viele Worte erklärt dies ein Beispiel.

Soll in der 7-Segment-Anzeige eine 6 angezeigt werden, so muß diese Zahl erst einmal in ein Bitmuster umgesetzt werden. Das geschieht im einfachsten Fall durch eine Umsetzung „im Kopf".

Dezimalzahl	Bitmuster	Sedezimalzahl
6	1Ø1Ø1111	AF

So muß im Grunde jede einzelne Zahl in ein Bitmuster umgesetzt werden. Weil im Programmablauf alles in Sedezimalzahlen programmiert wird, ist zusätzlich noch eine Codierung im Sedezimalsystem notwendig.

Die Funktion des Ports C

Eben haben Sie gesehen, wie eine einzelne Zahl auf einer 7-Segment-Anzeige dargestellt wird. In unserem kleinen Beispiel ist zur Darstellung der Zahl 6 die Ausgabe der Sedezimalzahl AFH auf den Port B erforderlich. Wenn Sie dies ausprobieren wollten, würde es nicht funktionieren. Das hat einfach den Grund, daß eine Anzeige erst eimal „aktiviert" werden muß, damit sie auch wie gewünscht arbeitet. Einfach ausgedrückt würde man sagen: Der Strom muß erst einmal eingeschaltet werden.

Diese Funktion übernimmt der Port C des 8255, und zwar die Bits Nr. Ø bis Nr. 5. Um die Zahl 6 auf der ganz links außen stehenden Anzeige darzustellen, ist es demnach notwendig, daß Sie zuerst auf den Port B die Sedezimalzahl AFH bringen, und außerdem auf den Port C die Sedezimalzahl ØlH. Erst dann ist die Anzeige aktiviert und die gewünschten Segmente leuchten auf. Wir werden dazu im gleich folgenden Kapitel Software einmal ein Programm schreiben.

Die Funktion des Ports A

Der Port A hat die Funktion, die Tastatur in Zusammenarbeit mit dem Port C abzufragen. Wie diese Arbeitsweise im einzelnen funktioniert, ist der Funktion der 7-Segment-Anzeige ähnlich. Nur wird in diesem Fall keine Zahl ausgegeben, vielmehr wird aus einem Tastendruck eine Sedezimalzahl gebildet. Diese Vorgangsweise wird in einem späteren Kapitel besprochen.

Die Betriebsart 1 des 8255

Bislang haben wir den 8255 in allen Programmen in der Betriebsart 0 verwendet. In dieser Betriebsart arbeitet der Ein-Ausgabebaustein ohne irgendwelche Anforderungs- oder Quittungssignale beim Datentransfer. Oft ist es aber wünschenswert, wenn nicht sogar unumgänglich, daß beim Datentransfer Steuersignale den Datenfluß überwachen und sozusagen Verkehrspolizei spielen.

Um eine neue Betriebsart des 8255 geht es in diesem Kapitel unseres Lehrgangs. Die Entwickler des 8255 haben dieser Betriebsart die Bezeichnung 1 gegeben. In der Betriebsart 1 gelangen Daten vom und zum Mikroprozessor über den Port A und den Port B. Beide Ports sind gleichberechtigt und können als Eingabeport oder als Ausgabeport arbeiten – je nachdem, welches Steuerwort sich im Steuerwortregister befindet. Die ein- oder ausgehenden Daten werden im Baustein zwischengespeichert. Bis hierher ist noch kein merklicher Unterschied zu der Ihnen nun schon zu genüge bekannten Betriebsart 0 festzustellen.

Das ändert sich jedoch, wenn man die Funktion des Ports C in dieser Betriebsart betrachtet. Die Leitungen des Ports C werden nun teilweise als Steuerleitungen für Port A und Port B verwendet. Jeder Port hat drei Leitungen des Ports C als Steuerleitungen zur Verfügung. Bleiben also noch 2 Bits des Ports C übrig. Diese beiden Leitungen können als Datenein- oder ausgabeleitungen in der Betriebsart 0 arbeiten. Die Bedeutungen der Steuerleitungen werden am einfachsten durch ein paar Beispiele klar. Gehen wir gleich in medias res.

Port A und Port B arbeiten als Eingabeports

Sehen Sie sich zunächst das Bild H76.1 an. Es zeigt schematisch die Funktion der einzelnen Ports des 8255, wenn die Ports A und B in der Betriebsart 1 arbeiten; und zwar als Eingabeports. Port C ist nun mit verschiedenen Steuersignalen belegt, deren Bedeutung gleich durchgesprochen werden soll.

Die Bedeutung der Steuersignale an Port C

Sobald Daten zur Eingabe am Port A anstehen, muß dem Ein- Ausgabebaustein 8255 mitgeteilt werden, daß er die Daten übernehmen soll. Dieser Vorgang wird durch die **Steuerleitung $\overline{\text{STB}}$** (**ST***ro***B***e* – Übernahme) eingeleitet. Mit einem Ø-Signal am Bit Nummer 4 des Ports C wird das Signal aktiviert. Mit der abfallenden Flanke dieses Signals werden die am Port A anstehenden Daten in einen internen Zwischenspeicher übernommen. Sowie die Daten im Zwischenspeicher angekommen sind, gibt der 8255 ein Quittungssignal an das Gerät ab, das die Daten angeliefert hat.

Dieses Quittungssignal wird an Bit Nummer 5 des Ports C abgegeben. Es hat die Bezeichnung **IBF** (**I***nput* **B***uffer* **F***ull* – Eingangsspeicher voll). Ist dieses Signal aktiv, so ist das ein Zeichen für das angeschlossene Gerät, daß es theoretisch die nächsten Daten anliefern kann.

Bis zu diesem Zeitpunkt ist die Mitwirkung des Mikroprozessors überhaupt nicht benötigt worden. Der Ein- Ausgabebaustein hat lediglich mit der „Peripherie“ korrespondiert – man könnte sagen: Endlich zeigt der Baustein einmal etwas Intelligenz und führt eine Tätigkeit selbst aus. Es ist fast so, als ob der Baustein der Sekretär des Mikroprozessors ist.

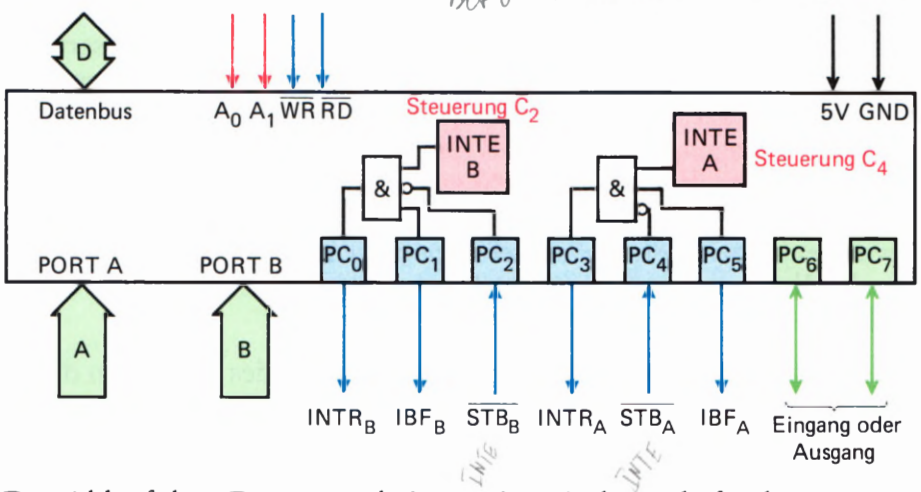


Bild H 76.1
Das Blockschaltbild des 8255 in der Betriebsart 1. Port A und Port B arbeiten als Eingabeports. Port C liefert die Steuersignale. Je drei Steuersignale stehen für jeden Port zur Verfügung.

Der Ablauf der „Datenverarbeitung“ ist mit dem Chef-Sekretär-Verhältnis am einfachsten zu erklären. Nehmen wir an, wichtige Unterlagen sollen zu einem Abteilungschef einer Firma gelangen. Wie sieht der Arbeitsablauf aus?

1. Zunächst wird der Anlieferer beim Sekretär des Abteilungschefs klingeln (entsprechend dem Strobe-Signal). Danach wird der Sekretär Einlaß gewähren und die wichtigen Unterlagen im Empfang nehmen (entsprechend dem Zwischenspeicher beim 8255). Was geschieht nun weiter?

2. Der Sekretär wird die Unterlagen ja an den Abteilungschef weitergeben wollen. Er klingelt also bei seinem Chef an und erbittet eine Unterbrechung seiner momentanen Tätigkeit.

3. Nun gibt es zwei Möglichkeiten. Entweder der Chef nimmt die Unterlagen an oder er sagt barsch: Jetzt nicht! Im ersten Fall gibt es keine Schwierigkeiten. Im zweiten Fall allerdings muß sich der Sekretär gedulden und zu einem anderen Zeitpunkt wieder nachfragen, ob der Chef nun Zeit hat. Dies muß solange geschehen, bis die Daten endgültig beim Abteilungschef angekommen sind. Genau der gleiche Vorgang spielt sich ab, wenn Daten in der Betriebsart 1 vom 8255 zum Mikroprozessor gelangen sollen.

Sie mögen uns diesen simplen Vergleich verzeihen, aber durch einfache Denkmodelle kann man sich manchmal einen sonst vielleicht unübersichtlichen Vorgang leichter vorstellen. Ab dem Arbeitsablauf unter Punkt 2 geschieht vergleichsweise in einem Mikroprozessorsystem folgendes:

Der 8255 gibt an den Mikroprozessor ein Signal ab, das dessen momentane Tätigkeit unterbrechen will: Es wird ein Interrupt angefordert.

Die Interrupt-Verarbeitung

Nun wird es etwas komplizierter. Die Daten sind im Zwischenspeicher angelangt; soweit waren wir schon. Nun muß ein Interrupt-Signal aktiviert werden. An dieser Stelle tritt die Frage auf, ob der Mikroprozessor sich überhaupt in seiner Tätigkeit unterbrechen lassen will. Von der Seite des Mikroprozessors gibt es zwei Möglichkeiten, denn wie jeder gute Chef kann er sich sogar durch zwei Vorkehrungen gegen Unterbrechungen absichern:

1. Er hat die Möglichkeit, durch Setzen oder Rücksetzen eines Flipflops (eines Signalspeichers) im 8255 diesem mitzuteilen, daß er nicht unterbrochen werden will. Dieses **Flipflop** im 8255 hat die Bezeichnung **INTE** und kann durch ein Steuerwort – ähnlich wie bei der Initialisierung des 8255 – bedient werden. Wie dies gemacht wird, erfahren Sie gleich im Fachgebiet Software dieses Lehrbriefs.

2. Der Mikroprozessor kann sich durch den Befehl DI (**D***isable* **I***nterrupt* – Unterbrechung nicht möglich) vor Unterbrechungen schützen.

Nimmt man einfach an, der Mikroprozessor sei für Unterbrechungen bereit, das heißt, das **INTE-Flipflop** hat 1-Pegel, dann gibt der 8255 am Bit Nummer 3 des Port C sein **INTR-Signal** (Unterbrechungs-Anforderung) ab. Sowie dieses Signal den Mikroprozessor erreicht, wird dieser seine Arbeit unterbrechen und eine Interrupt-Service-Routine abarbeiten. Dieser Programmteil muß den Mikroprozessor veranlassen, vom Zwischenspeicher des 8255 die Daten abzuholen und zu verarbeiten. Danach wird der Mikroprozessor seine vorige Tätigkeit wieder aufnehmen. Bild H77.1 zeigt noch einmal den groben Ablauf bei der Dateneingabe in der Betriebsart 1.

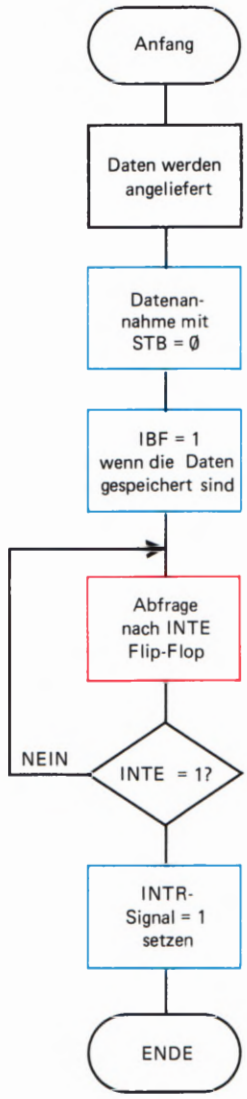


Bild H77.1
Der Programmablaufplan zeigt die einzelnen Phasen der Steuersignale in der Betriebsart 1. Dieses Programm ist natürlich in keinem Speicher abgelegt; es zeigt lediglich die Arbeitsweise des 8255.

Zusammenfassung der Steuersignale in der Betriebsart 1, Dateneingabe

Steuersignal	Port C	Bedeutung
$\overline{STB}_A$	PC_4	Anforderung an den 8255, Daten in seinen Zwischenspeicher zu laden. Die Anforderung kommt von Port A.
$\overline{STB}_B$	PC_2	Anforderung an den 8255, Daten in den Zwischenspeicher zu laden. Die Anforderung kommt von Port B.
IBF_A IBF_B	PC_5 PC_1	Quittungssignal des 8255, daß die Daten im Zwischenspeicher angekommen sind. Je nach Indizierung ist der Port A oder Port B gemeint.
$INTR_A$	PC_3	Interrupt-Anforderungssignal an den Mikroprozessor. Das Signal wird abgegeben, wenn das INTE A Flipflop gesetzt ist und IBF_A ebenfalls 1-Signal aufweist.
$INTR_B$	$PC_Ø$	Wie oben. Das Signal ist aktiv, wenn INTE B gesetzt und IBF_B 1-Signal hat.

H
78

Das Impulsdiagramm bei der Dateneingabe

Den zeitlichen Ablauf der Steuersignale bei einer Dateneingabe stellt man in der Technik häufig durch ein Impulsdiagramm dar. Auf diese Weise kann man sehen, was im Einzelnen geschieht, wenn Daten mit Hilfe eines Interrupts zum Mikroprozessor gelangen sollen. Das Bild H78.1 zeigt die vier wichtigsten Signale bei der Dateneingabe. Wenn Sie sich das Bild einmal ansehen, werden Sie feststellen, daß genau das wiedergegeben wird, was eben schon besprochen wurde.

Die Peripherie – also irgend ein an den 8255 angeschlossenes Gerät – meldet durch das **STB-Signal,** daß es Daten anliefert. Sowie dieses Signal Ø-Pegel hat, nimmt der 8255 die Daten in Empfang. Die Daten sind im Impulsdiagramm nicht dargestellt.

Sind die Daten im Zwischenspeicher angelangt, sendet der 8255 das **IBF-Signal** aus. Das wird im Bild H78.1 durch den roten Pfeil mit der Kennzeichnung 1 deutlich. Nun kann, wenn das interne **INTE-Signal** auf 1 steht, an den Mikroprozessor ein Interrupt gemeldet werden – das **INTR-Signal** nimmt 1-Pegel an. Das ist durch den mit der 2 gekennzeichneten roten Pfeil markiert.

Akzeptiert der Mikroprozessor den Interrupt, wird er in eine Programmrotuine springen, die ihn dazu veranlaßt, Daten vom 8255 abzuholen. Daß er das tut, wird dem 8255 durch Ø-Pegel am **$\overline{\text{RD}}$-Eingang** signalisiert – der 8255 nimmt als Folge davon sein **INTR-Signal** wieder zurück und meldet der angeschlossenen Peripherie, daß der Zwischenspeicher wieder „leer" ist. Das **IBF-Signal** wird durch das aufsteigende **$\overline{\text{RD}}$-Signal** wieder zurückgenommen und neue Daten können angeliefert werden; das Spiel kann erneut beginnen.

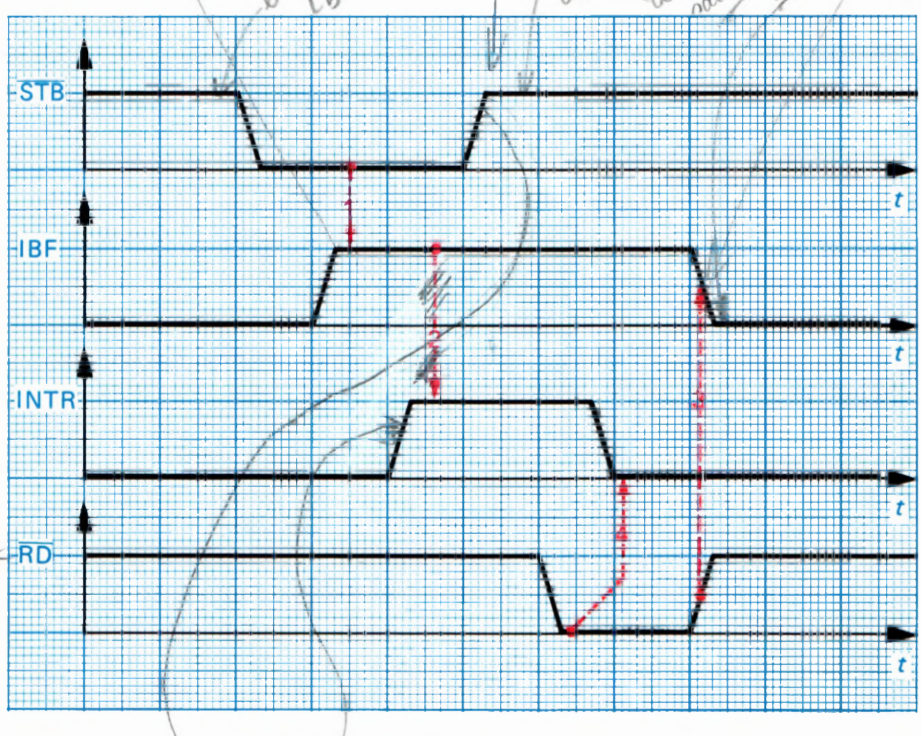


Bild H78.1
Das Impulsdiagramm für die Dateneingabe in der Betriebsart 1. Die Signale sind voneinander in bestimmter Weise abhängig. Die roten Pfeile zeigen diese Abhängigkeit.

Bildung verschiedener Steuerwörter in der Betriebsart 1

Nach dem Schema auf der Seite H 69.1 können Sie für die Betriebsart 1 des 8255 auf einfache Weise die verschiedenen Steuerwörter zusammenstellen. Wir haben dies in der nachfolgenden Tabelle zu Ihrer Arbeitserleichterung schon getan:

Steuerwort	Sedez.	Port A	Port B	freie Bits Port C
1Ø11 111X	BF	Eingang	Eingang	Eingang
1Ø11 Ø11X	B7	Eingang	Eingang	Ausgang
1Ø11 110X	BD	Eingang	Ausgang	Eingang
1Ø11 Ø10X	B5	Eingang	Ausgang	Ausgang
1Ø1Ø 111X	AF	Ausgang	Eingang	Eingang
1Ø1Ø Ø11X	A7	Ausgang	Eingang	Ausgang
1Ø1Ø 110X	AD	Ausgang	Ausgang	Eingang
1Ø1Ø Ø1ØX	A5	Ausgang	Ausgang	Ausgang

Das Bit Nummer 1 entscheidet jeweils über die Funktion des Ports B. Bit Nummer 3 entscheidet über die nicht für Steuerzwecke benutzten Bits des Ports C. Bit Nummer 5 schließlich ist ausschlaggebend für die Funktion des Ports A. Das mit X bezeichnete Bit Nummer Ø haben wir mit einer 1 belegt.

Sie sehen, auch in dieser Betriebsart gibt es eine Menge von verschiedenen Kombinationen der Portfunktionen. Doch sehen wir uns nun noch eine markante Kombination des Ports A und Ports B an:

Port A und Port B arbeiten als Ausgabeports in Betriebsart 1

Wenn die beiden Ports als Ausgabeports arbeiten, ändern sich die Steuersignale, die für die Betriebsart 1 notwendig sind. Es sind im Grunde auch wieder Signale, die den Datenverkehr zwischen der Peripherie – also den Geräten, die an das Mikroprozessorsystem angeschlossen sind – und dem Mikroprozessor steuern. Auch diese Steuersignale werden am Port C abgegriffen. Wie schon bei der eben besprochenen Dateneingabe, stehen auch für die Datenausgabe wieder für jeden Port drei Steuersignale zur Verfügung.

Die Steuersignale haben jetzt allerdings, bedingt durch die geänderte Datenflußrichtung, andere Bedeutungen. Ebenfalls ist die Reihenfolge der Signale bei dieser Art des Datentransfers anders als bei der Dateneingabe. Gehen wir zunächst einmal durch, was bei der Datenausgabe im einzelnen geschehen muß.

Die Bedeutung der Steuersignale an Port C

Daten sollen nun vom Mikroprozessor zur Peripherie gelangen. So ist es nicht verwunderlich, wenn bei der Datenausgabe nun zuerst der Mikroprozessor aktiv werden muß. Er liefert seine Daten zunächst in den Zwischenspeicher des 8255; je nachdem auf welchen Port die Ausgabe erfolgen soll, eben in den Zwischenspeicher für Port A oder Port B.

Sind die Daten im Zwischenspeicher des 8255 angelangt, gibt der Baustein an die Peripherie ein Signal ab, das **$\overline{\text{OBF}}$** (**O***utput* **B***uffer* **F***ull* – Ausgangsspeicher voll) genannt wird. Dieses Signal ist ein Anforderungssignal an die Peripherie, daß Daten abholbereit im Zwischenspeicher des 8255 anstehen.

Nun kann die Peripherie reagieren. Als Zeichen, daß die im Zwischenspeicher stehenden Daten abgeholt wurden, muß am **$\overline{\text{ACK}}$**-Eingang (**ACK***nowledge*-Quittungssignal) ein 0-Pegel angelegt werden. Die absteigende Flanke dieses Signals veranlaßt den 8255 das **$\overline{\text{OBF}}$**-Signal zurückzunehmen – als Zeichen dafür, daß der Zwischenspeicher wieder leer ist. Mit der aufsteigenden Flanke des **$\overline{\text{ACK}}$**-Signals gibt die Peripherie bekannt, daß die Daten übernommen wurden. Mit dieser Flanke wird auch das Interrupt-Signal **INTR** aktiv, wenn das interne **INTE**-Flipflop gesetzt ist. Das Interrupt-Signal kann den Mikroprozessor dazu veranlassen, einen erneuten Schreibvorgang auszuführen.

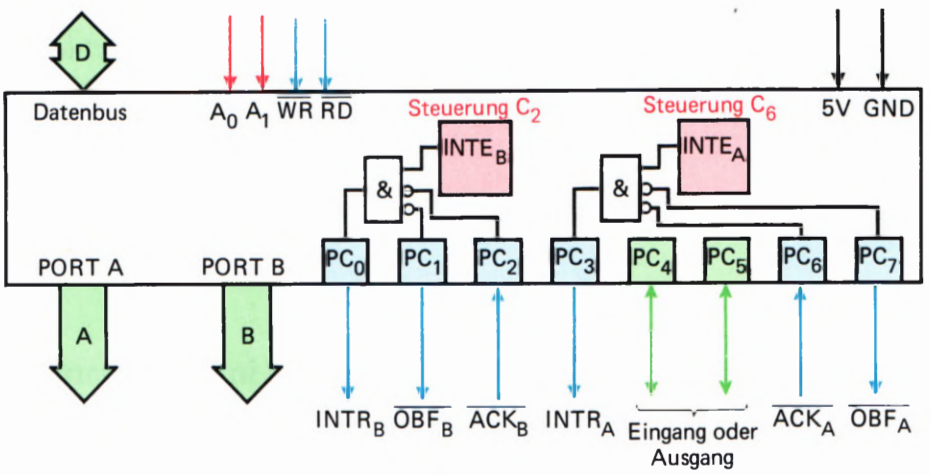


Bild H 80.1
Das Blockschaltbild für die Datenausgabe in der Betriebsart 1 des 8255. Beide Ports arbeiten als Ausgabeports. Port C liefert die entsprechenden Steuersignale.

Bleibt für den 8255 nur noch eines zu tun: Das Interrupt-Signal wieder zurückzunehmen. Dies geschieht durch die aufsteigende Flanke des **$\overline{\text{WR}}$-Signals;** also dann, wenn der Mikroprozessor die nächsten Daten an den Zwischenspeicher des 8255 liefert. Damit ist der Auslesevorgang von Daten an die Peripherie auch schon zu Ende.

Bild H 80.1 zeigt die Bedeutung der Signale am Port C, wenn beide Ports als Ausgabeports arbeiten.

Das Impulsdiagramm bei der Datenausgabe

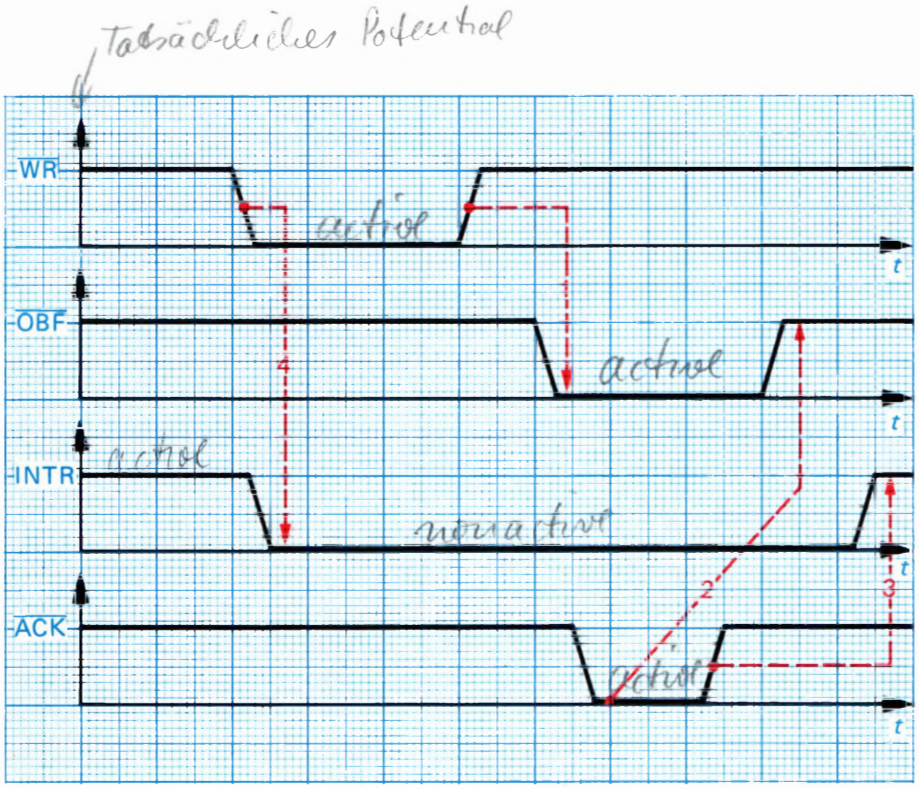
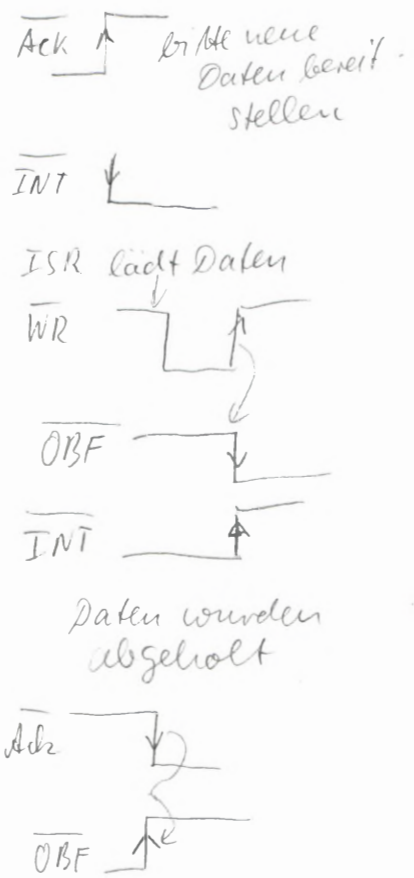


Bild H 81.1
Das Impulsdiagramm für den zeitlichen Ablauf der Steuersignale in der Betriebsart 1. Die Abhängigkeit der einzelnen Signale voneinander ist durch die roten Pfeile dargestellt.



Das Impulsdiagramm in Bild H 81.1 zeigt den zeitlichen Ablauf der Steuersignale, wenn ein Port des 8255 als Ausgabeport in der Betriebsart 1 arbeitet. Die Zusammenfassung der einzelnen Signale in Kurzfassung:

Das $\overline{\text{WR}}$-Signal wird aktiv (mit Ø-Pegel), wenn der Mikroprozessor Daten in den Zwischenspeicher des 8255 schreibt. Die absteigende Flanke dieses Signals setzt das **Interrupt-Signal INTR** zurück, als Zeichen dafür, daß neue Daten in den Zwischenspeicher des 8255 gelangt sind. Damit hat die Peripherie die Möglichkeit, einen erneuten Interrupt anzufordern. Die aufsteigende Flanke des **$\overline{\text{WR}}$-Signals** aktiviert das **$\overline{\text{OBF}}$-Signal,** das anzeigt, daß Daten im Zwischenspeicher abholbereit sind.

Das $\overline{\text{OBF}}$-Signal wird, wie schon gesagt, durch die aufsteigende Flanke des **$\overline{\text{WR}}$-Signals** aktiviert. Aktiv ist dieses Signal durch einen Ø-Pegel, das wird auch durch den Querbalken über der Bezeichnung deutlich. Das **$\overline{\text{OBF}}$-Signal** gibt der an den 8255 angeschlossenen Peripherie bekannt, daß Daten vom Zwischenspeicher abgeholt werden können. Aber das wissen Sie ja bereits.

Das $\overline{\text{ACK}}$-Signal ist ein Quittungssignal der Peripherie, daß die Daten, die am entsprechenden Port anstehen, abgeholt werden. Auch dieses Signal ist durch einen Ø-Pegel aktiv. Mit der absteigenden Flanke wird das **$\overline{\text{OBF}}$-Signal** zurückgenommen. Durch die aufsteigende Flanke des Signals kann die Peripherie einen Interrupt auslösen, wenn das interne **INTE-Flipflop** gesetzt ist. Wie das interne INTE-Flipflop gesetzt wird, erfahren Sie im Fachgebiet Software des Lehrgangs. Bleibt nur noch das **INTR-Signal** genauer anzusehen.

Das INTR-Signal löst einen Interrupt am Mikroprozessor aus und wird durch die aufsteigende Flanke des $\overline{ACK}$-Signals gesetzt. Rückgesetzt wird das Signal durch die abfallende Flanke des $\overline{WR}$-Signals. Halten Sie sich den Arbeitsablauf noch einmal kurz vor Augen:

Zuerst liefert der Mikroprozessor die Daten an den Zwischenspeicher des 8255. Danach werden die Daten von der Peripherie vom jeweiligen Port abgeholt. Dieses Abholen bedeutet für den Mikroprozessor, daß er erneut Daten liefern kann. Ein typisches Beispiel für diese Art des Datentransfers ist das Übersenden von Daten an einen Drucker. Der Drucker kann durch das $\overline{ACK}$-Signal anzeigen, daß er Daten abholt; und das Mikroprozessor-System kann durch das $\overline{OBF}$-Signal angeben, wenn neue Daten abholbereit sind. Ohne die Betriebsart 1 wäre diese Art von Datentransfer kaum oder nur sehr umständlich möglich. Bild H 82.1 zeigt einen möglichen Anschluß eines Druckers an den 8255.

Die eben besprochenen Datenflußrichtungen der einzelnen Ports sind natürlich ganz individuell mischbar. Das heißt, daß selbstverständlich zum Beispiel der Port A als Ausgabeport arbeiten kann, während der Port B als Eingabeport arbeitet oder umgekehrt. Entscheidend für die Arbeitsweise der einzelnen Ports ist lediglich das einmal in der Steuerwortregister eingeschriebene Steuerwort. Wie dieses gebildet wird, haben Sie ja schon dem Schema auf der Seite H 69 entnommen. Eine Zusammenstellung der Steuerwörter, die für die Betriebsart 1 wichtig sind, finden Sie auf der Seite H 79.

Wenn Sie Ein- und Ausgaben in der Betriebsart 1 machen möchten, sollten Sie darauf achten, welche Steuersignale an welchen Bits des Ports C ausgegeben bzw. eingelesen werden.

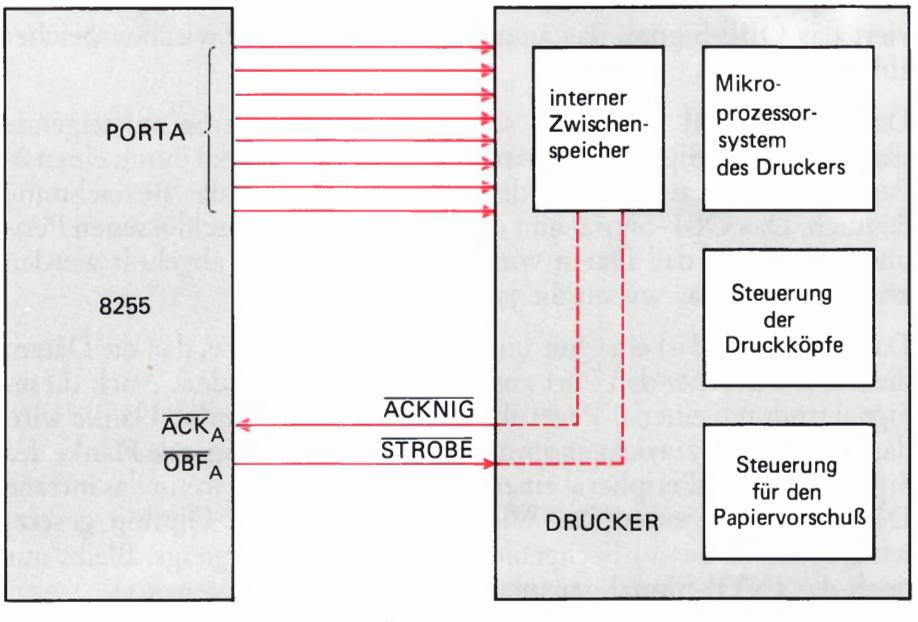


Bild H 82.1
Ein Anwendungsbeispiel der Betriebsart 1 ist die Ansteuerung eines Druckers. Die Daten werden hier jeweils 8-Bit-parallel angeliefert. Über die beiden Steuersignale wird bekanntgegeben, ob die Daten angekommen sind bzw. ob sie abgesandt werden sollen.

Die Betriebsart 2 des 8255

Das letzte Kapitel dieses Lehrgangs berichtet über die Betriebsart 2 des 8255. Diese Betriebsart ist zumindest von der Bedeutung der Steuersignale her der Betriebsart 1 nahezu identisch. Was ändert sich also? Nun, der große Unterschied zwischen der Betriebsart 1 und der Betriebsart 2 ist der, daß in der Betriebsart 1 immer festgelegt sein muß, ob Port A als Eingabeport oder als Ausgabeport arbeiten soll. In der Betriebsart 2 dagegen kann der Port A des 8255 sowohl als Eingabeport als auch als Ausgabeport arbeiten, ohne daß dafür das Steuerwort geändert werden muß.

Je nachdem, welche Steuersignale bedient werden, stellt sich der Port A des Bausteins auf die gewünschte Datenflußrichtung selbst ein. Bild H 83.1 zeigt das Blockschaltbild des Bausteins in der Betriebsart 2. Wenn Sie sich dieses Bild ansehen, fällt zunächst auf, daß die Steuersignale etwas anders verknüpft sind als in der Betriebsart 1. Das Interrupt-Signal zum Beispiel wird nun von jeweils zwei Steuersignalen bedient. Aber gehen wir auch hier lieber Schritt für Schritt vor.

Die Bedeutung der Steuersignale an Port C

Port C liefert auch in der Betriebsart 2 die jeweils geforderten Steuersignale für die Dateneingabe oder die Datenausgabe. Sollen Daten eingelesen werden, muß wie in der Betriebsart 1 zunächst von der Peripherie ein **$\overline{\text{STB}}$**-Signal geliefert werden. Dieses Signal zeigt an, daß Daten zum Einlesen bereitstehen. Sowie der 8255 diese Daten in seinen Zwischenspeicher eingelesen hat, gibt er dies durch das **IBF**-Signal bekannt.

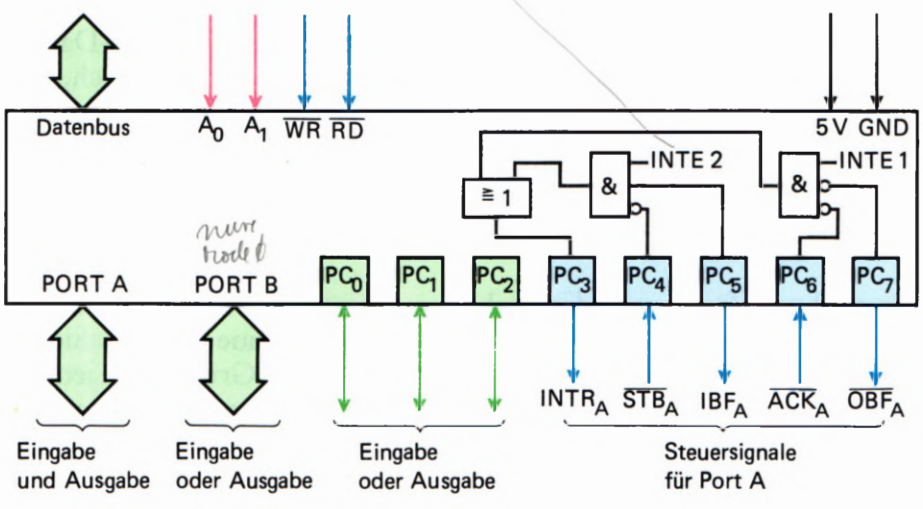


Bild H 83.1
Der Blockschaltplan des 8255 für die Betriebsart 2. Der Port A arbeitet als Zweiwegbus. Port B kann in dieser Betriebsart nur noch in Betriebsart 0 betrieben werden.

Wenn das interne INTE-Flipflop gesetzt ist, gibt der Baustein an den Mikroprozessor ein Interrupt-Signal ab, damit dieser die Daten aus dem Zwischenspeicher abholt. Sie sehen, es ist alles so, wie wir es schon bei der Betriebsart 1 besprochen haben.

Auch die Arbeitsweise bei der Datenausgabe wird Ihnen bekannt vorkommen. Werden Daten vom Mikroprozessor zum 8255 zur Datenausgabe angeliefert, sendet der Baustein das $\overline{\text{OBF}}$-Signal aus; ein Zeichen für die Peripherie, daß Daten abholbereit am Port A anstehen. Daß die Daten abgeholt werden, wird dem 8255 durch das $\overline{\text{ACK}}$-Signal bekanntgegeben. Sind beide Signale aktiv und das zugehörige interne INTE-Signal gesetzt, kann der 8255 ein Interrupt-Signal an den Mikroprozessor aussenden. Das Interrupt-Signal ist also für beide Datenflußrichtungen zuständig. Das ist ebenfalls ein neuer Aspekt beim Arbeiten in dieser Betriebsart.

Die internen INTE-Flipflops

In der Betriebsart 1 stand für jeden Port ein INTE-Flipflop zur Freigabe oder Sperrung eines Interrupt-Signals zur Verfügung. In der Betriebsart 2 können Daten sowohl nach „außen" als auch nach „innen" gehen. Aus diesem Grund wird es notwendig, für den Port A zwei INTE-Flipflops zur Verfügung zu stellen. Sie werden in der Betriebsart 2 durch die Leitungen C4 und C6 bedient. Folgende Zusammenstellung gibt Ihnen wieder eine Übersicht, wie diese Flipflops gesetzt oder rückgesetzt werden.

Betriebsart 2		Steuerwort für	
INTE 1	INTE 2	INTE 1	INTE 2
C6 Ausgabe	C4 Eingabe	Ø111 110X	Ø111 100X

1 = setzen
Ø = rücksetzen

Der Port B und die freien Bits des Ports C

Alle Steuersignale des Ports C werden nun für die Zweiweg-Datenrichtung des Ports A gebraucht. Somit ist es nicht verwunderlich, daß der Port B nur noch in der Betriebsart 0, als Datenausgabe- oder Dateneingabeport arbeiten kann. Dies trifft auch für die nicht benutzten Bits des Ports C zu.

Sie sind nun fast am Ende des Lehrgangs angekommen, und haben schon eine ganze Menge über Peripherie-Bausteine gelernt. Sicher bedarf es jetzt noch einiger Programmübungen, die Grundlagen jedoch haben Sie „intus". Wenn Sie Ihr neu gelerntes Wissen überprüfen wollen, lösen Sie jetzt bitte die gestellten Prüfungsaufgaben. Wir wünschen Ihnen dabei viel Erfolg.

Lehrgang PERIPHERIE-BAUSTEINE Verfasser: Hans Fischer, Edgar Hoch
Herausgeber: R. Christiani

Software

Die Programmierung der Z 80 PIO

Wir haben bereits darauf hingewiesen, daß die Port-Anschlüsse des Schnittstellen-Bausteins Z 80 PIO sowohl die Funktion eines Daten-Senders als auch die eines Daten-Empfängers übernehmen können. (Vgl. Seite H 15.) Der Schnittstellen-Baustein kann also als Sender von der CPU gelieferte Daten an eine Peripherie übermitteln; er kann aber auch von der Peripherie gelieferte Daten empfangen und diese an die CPU weiterleiten.

Es kann nicht das Ziel unseres Lehrgangs sein, die interne Schaltung eines Schnittstellen-Bausteins zu analysieren. Ähnlich wie die interne Schaltung einer CPU ist diese Schaltung sehr komplex und letztlich für den Anwender auch gar nicht interessant. Aber auch ohne die Kenntnis der Schaltung ist leicht einzusehen, daß beim Senden und beim Empfangen von Daten im Inneren des Bausteins recht unterschiedliche Schaltungs-Strukturen tätig werden müssen.

Im Zusammenhang mit einer bestimmten Aufgabe, die ein Mikroprozessor-System lösen soll, wird man an einen Port eines Schnittstellen-Bausteins eine ganz bestimmte Peripherie, z. B. einen Drucker oder eine Tastatur, anschließen. Es ist sehr unwahrscheinlich, daß man gerade dann, wenn das Mikroprozessor-System eine bestimmte Aufgabe löst, plötzlich die Tastatur gegen z. B. irgendeine optische Anzeige austauschen möchte.

Eine Tastatur ist ein Daten-Sender. Von ihr empfängt der Schnittstellen-Baustein Daten. Der Port des Bausteins, an den die Tastatur angeschlossen ist, muß also als Daten-Empfänger arbeiten. Würde man an den gleichen Port statt der Tastatur eine optische Anzeige anschließen, dann müßte dieser Port die Funktion eines Daten-Senders annehmen, also intern andere Schaltungs-Strukturen aktivieren.

Diese Überlegungen zeigen, daß man die Einstellung eines Ports des Schnittstellen-Bausteins zum Senden oder zum Empfangen von Daten jeweils vor dem Lösen einer Aufgabe vornehmen wird.

Einer der großen Vorteile solcher Schnittstellen-Bausteine ist es, daß diese Einstellung keine Sache der Hardware ist: Der Lötkolben kann also kalt bleiben. Die Einstellung wird per Software vorgenommen; der Baustein kann programmiert werden.

Die Möglichkeit, einen Schnittstellen-Baustein programmieren zu können, unterstreicht unsere Feststellung auf der Seite H 8: Der Baustein ist eine Art Mikroprozessor, der für die Lösung ganz bestimmer Aufgaben konzipiert wurde. Wegen dieser Spezialisierung ist der „Befehlssatz“ eines solchen Bausteins wesentlich kleiner als der einer normalen CPU. Man verzichtet auch meist auf eine mnemonische Kennzeichnung der Befehle. Trotzdem gibt es auch für Schnittstellen-Bausteine Ein-, Zwei- und Drei-Byte-Befehle.

Die Programmierung eines Schnittstellen-Bausteins wird man meist am Beginn des Programms vornehmen, mit dem das Mikroprozessor-System eine bestimmte Aufgabe lösen soll. Das Programm muß also Anweisungen an die CPU enthalten, dem Schnittstellen-Baustein nacheinander eine Reihe von Befehls-Bytes zu übermitteln.

S
2

Es stellt sich natürlich sofort die Frage, wie der Schnittstellen-Baustein feststellen soll, ob ihm von der CPU gerade Befehls-Bytes zu seiner Programmierung übermittelt werden, oder ob es sich bereits um Daten-Bytes handelt, die er an die angeschlossene Peripherie weitergeben soll. Wir werden Ihnen im folgenden Abschnitt zeigen, wie diese Unterscheidung getroffen werden kann.

Zunächst gehen wir davon aus, daß der Schnittstellen-Baustein die ihm übermittelten Bytes als Befehls-Bytes erkennt. Wir stellen Ihnen diese Befehls-Bytes der Reihe nach vor. Dabei werden Sie erkennen, was man denn dem Schnittstellen-Baustein überhaupt befehlen kann. Man kann ihm nämlich wesentlich mehr befehlen als die einfache Umschaltung eines Port von Empfangen auf Senden oder umgekehrt.

Die Betriebsarten der Z 80 PIO

Es ist Ihnen sicher kein Geheimnis, daß die ganze Mikroprozessor-Sippschaft amerikanischen Ursprungs ist und auch wenig Neigung zeigt, diesen Ursprung zu verleugnen. Am deutlichsten wird das an der Sprache. Alles, was mit Mikroprozessoren zu tun hat, will grundsätzlich englisch ausgesprochen werden: Sämtliche Mnemonics der CPU-Befehle z. B. leiten sich selbstverständlich aus dem Englischen ab. Aber auch fast alle anderen Bezeichnungen im Bereich der Mikroprozessoren stammen aus der englischen Sprache.

Wir weisen auf diese Tatsache hin, weil das, was wir hier mit "Betriebsart" bezeichnen, in den Datenblättern der Z 80 PIO und bei fast allen Anwendungen dieses Bausteins *Mode* (sprich: moud) genannt wird.

Die Ports der Z 80 PIO können in vier unterschiedlichen Betriebsarten (vier unterschiedlichen Modes) arbeiten. Zu Beginn eines Programms, bei dessen Ablauf die CPU über die Z 80 PIO mit einer Peripherie Daten austauscht, muß dem Schnittstellen-Baustein per Befehls-Byte mitgeteilt werden, welche Betriebsart für den verwendeten Port gewünscht wird. In der Fachsprache wird dieses Betriebsarten-Befehls-Byte als *Mode Control Word* (Betriebsart-Steuerwort) oder einfach als *Mode Word* bezeichnet.

In seiner einfachsten Form wird das Betriebsart-Steuerwort von der CPU an die Z 80 PIO als Ein-Byte-Befehl übermittelt:

```
LD   A,ModeWord     ;Das Steuerwort wird in den Akku geladen
OUT  (Port),A       ;  und an die Z 80 PIO geliefert.
```

(Um welchen Port es sich bei dem OUT-Befehl handelt, erläutern wir im nächsten Abschnitt.)

Was bewirkt nun dieses Betriebsart-Steuerwort *(Mode Control Word)*, wie sieht es aus, und was hat es mit den vier unterschiedlichen Betriebsarten überhaupt auf sich?

Wir wollen vorausschicken, daß – mit einer eigens zu erwähnenden Ausnahme – jeder Port der Z 80 PIO unabhängig vom anderen Port für eine Betriebsart programmiert werden kann. Wenn bei einer bestimmten Aufgabenstellung beide Ports der Z 80 PIO verwendet werden, dann müssen dem Schnittstellen-Baustein zwei (u. U. unter-

schiedliche) Betriebsart-Steuerworte nacheinander übermittelt werden. Wohin diese Steuerworte geschickt werden müssen, ist eine Sache der Hardware, und das wird im nächsten Abschnitt beschrieben.

Wir stellen Ihnen hier zunächst die vier verschiedenen Betriebsarten vor. Anschließend werden wir dazu den einen oder anderen Versuch beschreiben.

Betriebsart 0 *(Mode 0)*

Das Steuerwort für diese Betriebsart programmiert den zugehörigen Port als **Sender,** der von der CPU übergebene Daten an eine Peripherie liefert.

Sobald an den acht Port-Leitungen ein Byte bereit steht, das von der Peripherie übernommen werden kann, schaltet der Schnittstellen-Baustein auf den zum Port gehörenden READY-Anschluß (sprich: räddie) ein 1-Signal. Dieses **READY**-Signal kann die Peripherie veranlassen, das bereitgestellte Byte abzuholen.

Wenn die Peripherie das Byte vom Port abgeholt hat, kann sie ihrerseits diese Tatsache mit einem kurzzeitigen 0-Signal an einen weiteren, zum Port gehörenden Anschluß quittieren. Dieses Quittungs-Signal wird als **STROBE** (sprich: stroub) bezeichnet. Die 0-1-Flanke am Ende des STROBE-Signals löscht das READY-Signal des Schnittstellen-Bausteins. Dadurch wird vermieden, daß die Peripherie das gleiche Byte zweimal liest.

Das STROBE-Signal kann den Schnittstellen-Baustein veranlassen, von der CPU das nächste Byte für die Peripherie bereitstellen zu lassen.

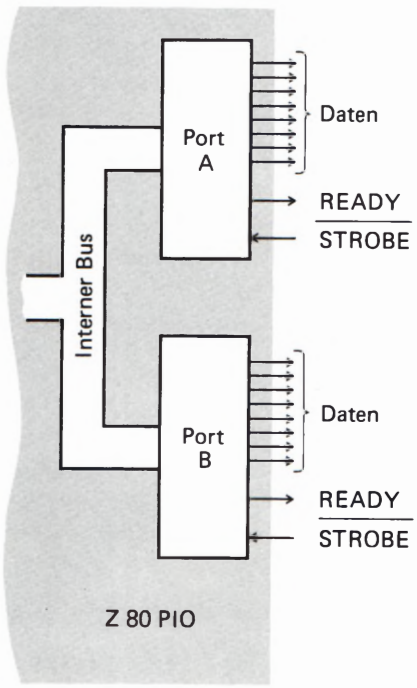


Bild S3.1
Anschlüsse der Ports A und B an der Z80 PIO.

Hier haben wir zwei Port-Anschlüsse erwähnt, von denen bisher noch nicht die Rede war. In die Prinzip-Darstellung im Bild H12.1 haben wir nur jeweils acht zu einem Port gehörende Anschlüsse eingetragen. Es sind die Anschlüsse für die acht Bits eines Bytes. Das ist auch richtig so, denn auf der Peripherie-Leiterplatte sind die Anschlüsse für die READY- und STROBE-Signale nicht zugänglich.

Das Bild S3.1 zeigt das vollständige Anschluß-Schema der beiden Ports A und B der Z80 PIO. Die Pfeile an den READY-Anschlüssen machen deutlich, daß es sich um Sender-Anschlüsse handelt, über die der Peripherie mit einem 1-Signal mitgeteilt wird, daß an den zugehörigen Port-Anschlüssen ein Byte zum Abholen bereit steht.

Die Pfeile an den STROBE-Anschlüssen zeigen, daß es sich um Empfänger-Anschlüsse handelt. – In unserer Darstellung haben wir das Wort STROBE mit einem Querstrich überstrichen. Damit wird angedeutet, daß dieses Signal *active LOW* ist: Es hat normalerweise den Signalwert 1 und meldet mit einem 0-Signal von der Peripherie, daß das zuletzt angelieferte Byte übernommen worden ist.

Der Melde- und Quittier-Betrieb bei der Daten-Komunikation zwischen einem Mikroprozessor-System und der zugehörigen Peripherie wird häufig verwendet. Man bezeichnet ihn sehr bildhaft als **Handshake**-Betrieb (sprich: händschäik; Händeschütteln.)

S
4

Betriebsart 1 *(Mode 1)*

Das Steuerwort für diese Betriebsart programmiert den zugehörigen Port als **Empfänger,** der von der Peripherie gesendete Daten an die CPU weiterleitet.

Wenn die CPU bereit ist, Daten von der Peripherie zu übernehmen, dann wird auf den READY-Anschluß des betreffenden Ports ein 1-Signal geschaltet. Die Peripherie kann dieses READY-1-Signal zum Anlaß nehmen, jetzt dem Port ein Byte zum Einlesen zur Verfügung zu stellen. Wenn das geschehen ist, dann kann die Peripherie diese Tatsache mit einem STROBE-0-Signal an den STROBE-Anschluß der Z 80 PIO melden.

Die Z 80 PIO speichert daraufhin das eingelesene Byte und schaltet anschließend das READY-Signal ab, so daß die Peripherie zunächst kein neues Byte an die Schnittstelle schickt.

Jetzt kann die CPU das von der Z 80 PIO eingelesene Byte abholen. Erst wenn das geschehen ist, gibt die Z 80 PIO wieder ein READY-1-Signal aus, und die Peripherie kann das nächstfolgende Byte anliefern.

Für die Betriebsart 1 ist – wie für die Betriebsart 0 – die Darstellung im Bild S 3.1 zuständig, mit dem Unterschied, daß Sie sich die Pfeile an den Daten-Anschlüssen der Ports in umgekehrte Richtung weisend vorstellen müssen.

Interessant ist in diesem Zusammenhang die Frage, wie die CPU erfährt, daß die Z 80 PIO ein Byte von der Peripherie eingelesen und zum Abholen durch die CPU bereitgestellt hat.

Es wäre denkbar, daß die CPU jeweils in einer Schleife bei der Z 80 PIO nachsieht, ob diese inzwischen ein neues Byte eingelesen hat. Tatsächlich bietet das Z 80-System eine wesentlich elegantere Möglichkeit an, die wir hier zunächst nur andeuten wollen. Sobald nämlich die Peripherie die Übertragung eines neuen Bytes mit dem STROBE-0-Signal an die PIO meldet, schickt die PIO an die CPU ein **Interrupt-Signal.** (Vgl. Lehrgang Mikroprozessortechnik, Seite S 9). Dieses Signal bewirkt eine Unterbrechung des gerade laufenden Programms und den Ansprung einer **Interrupt-Service-Routine,** in der das Abholen und die Verarbeitung des gerade eingelesenen Bytes erledigt wird. Wenn das geschehen ist, dann wird die Bearbeitung des gerade laufenden Programms wieder aufgenommen.

Wir werden Ihnen das Arbeiten mit Interrupt im Laufe dieses Lehrgangs noch eingehender vorstellen.

Die Betriebsart 2 bildet die bereits angekündigte Ausnahme bei der Programmierung der Ports:

Betriebsart 2 *(Mode 2)*

Das Steuerwort für die Betriebsart 2 programmiert den Port A als **Sender und** als **Empfänger.** Es können also sowohl Daten vom Mikroprozessor-System an die Peripherie geschickt als auch von der Peripherie an das System geschickte Daten eingelesen werden.

Beim Arbeiten in dieser Betriebsart kann nur der Port A verwendet werden. Der Port B muß für die Betriebsart 3 programmiert werden. Seine Daten-Anschlüsse werden abgeschaltet; verwendet werden die Handshake-Leitungen des Ports B. – Das Bild S 5.1 verdeutlicht die Arbeitsweise der Betriebsart 2.

Die Handshake-Leitungen des Ports A steuern die Daten-Ausgabe. Über den READY-Anschluß wird der Peripherie gemeldet, daß für die Übermittlung ein Byte bereitgestellt wurde. Wenn die Peripherie dieses Byte abgeholt hat, dann muß sie es dem Schnittstellen-Baustein an den STROBE-Anschluß des Ports A melden.

Wenn die CPU das Einlesen eines Bytes über den Port A von der Peripherie verlangt, dann setzt der Schnittstellen-Baustein ein 1-Signal auf die READY-Leitung des Ports B. Die Peripherie muß für diesen Fall so eingerichtet sein, daß sie dieses Signal erkennt. Sie liefert ein Byte an den Port A des Schnittstellen-Bausteins und meldet das mit einem 0-Signal an den STROBE-Anschluß des Ports B.

Wie bei der Z 80 PIO die Daten-Anschlüsse des Ports B abgeschaltet werden können, zeigen wir Ihnen bei der Vorstellung der Programmier-Befehle.

Die Betriebsart 2 setzt voraus, daß auch die Peripherie Anschlüsse für bidirektionalen Daten-Transport besitzt, über die also Daten wahlweise in der einen oder in der anderen Richtung geschickt werden können. Außerdem muß sie über die entsprechenden Handshake-Leitungen verfügen. Diese Betriebsart ist Spezialfällen vorbehalten.

Am universellsten verwendbar ist die Betriebsart 3, die wir auch in unseren Versuchen vorzugsweise benutzen wollen.

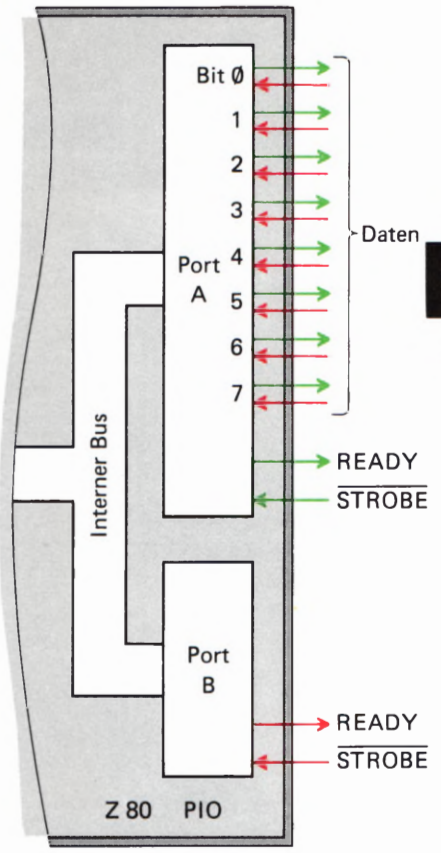


Bild S 5.1
In der Betriebsart 2 der Z 80 PIO können über die Anschlüsse des Ports A Daten ausgegeben und eingelesen werden. Vom Port B werden nur die Handshake-Anschlüsse benutzt.

Betriebsart 3 *(Mode 3)*

Die Betriebsart 3 kann sowohl für den Port A als auch für den Port B programmiert werden, unabhängig davon, ob der jeweils andere Port für eine der Betriebsarten 0, 1 oder 3 programmiert wurde.

Im Anschluß an das Steuerwort für die Betriebsart 3 müssen der Z 80 PIO einige weitere Befehle übermittelt werden, mit denen die Daten-Anschlüsse **Bit-weise** nach Wunsch als **Sender** oder als **Empfänger** programmiert werden können.

Die Handshake-Anschlüsse sind in der Betriebsart 3 nicht in Betrieb. Dafür kann man jeden der acht Daten-Anschlüsse des Ports nach Bedarf sozusagen als STROBE-Anschluß definieren. Wahlweise mit einem 0- oder mit einem 1-Signal kann über einen so programmierten Anschluß ein Interrupt-Signal ausgelöst werden, das die CPU spontan zu einer (natürlich vorher programierten) Aktion veranlaßt. (Vgl. Seite S 4.)

Die Programmierung der Interrupt-auslösenden Anschlüsse kann so vorgenommen werden, daß der Interrupt immer dann ausgelöst wird, wenn nur einer der so programmierten Anschlüsse mit einem 0- oder mit einem 1-Signal einen Interrupt anfordert. Es kann aber

auch so programmiert werden, daß ein Interrupt nur ausgelöst wird, wenn an allen entsprechend programmierten Anschlüssen gleichzeitig eine Interrupt-Anforderung gemeldet wird.

Diese kurze Zusammenstellung zeigt, daß die Betriebsart 3 außerordentlich vielseitig ist. Im einfachsten Fall kann die Programmierung so vorgenommen werden, daß alle Daten-Anschlüsse des auf diese Betriebsart programmierten Ports als Ausgänge oder auch als Eingänge definiert werden. Die Betriebsart 3 ähnelt dann der Betriebsart 0 bzw. der Betriebsart 1, allerdings ohne die Möglichkeiten des Handshakes.

Vorzugsweise wird man die Betriebsart 3 verwenden, wenn mit der Peripherie keine kompletten Bytes ausgetauscht werden sollen, sondern nur einzelne Bits. Das ist z. B. dann der Fall, wenn die Peripherie eine ganz andere Struktur als Daten-verarbeitende Geräte aufweist. Daten-verarbeitende Geräte sind meist auf die Verarbeitung von Bytes mit jeweils 8 Bits eingerichtet. Zur Steuerung einer irgendwie sonst gearteten Einrichtung braucht man aber einzelne Steuer-Signale und bekommt einzelne Signale zurückgemeldet.

Das Betriebsart-Steuerwort *(Mode Control-Word)*

Auf der Seite S2 haben wir Ihnen die zwei Befehle angegeben, mit denen der Z80 PIO das dort als ModeWord bezeichnete Betriebsart-Steuerwort übermittelt wird. Es handelt sich um einen Ein-Byte-Befehl für die PIO, dem allerdings – abhängig von der gewählten Betriebsart – im allgemeinen ein oder mehrere weitere Befehle zur vollständigen Programmierung der Z80 PIO folgen müssen.

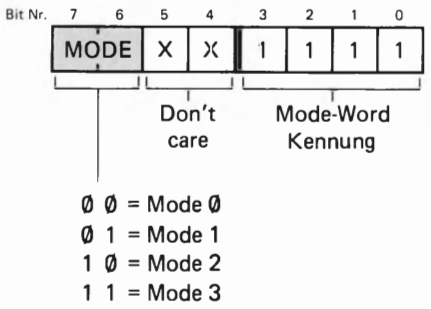


Bild S6.1
Aufbau des Betriebsart-Steuerworts *(Mode Control-Word)*.

Das Bild S6.1 zeigt den Bit-weisen Aufbau des Betriebsart-Steuerworts. – Die 1-Bits Nr. 0 bis Nr. 3 kennzeichnen das Byte für die PIO als *Mode-Word*. Wenn sie in einem Befehl diese vier 1-Bits vorfindet, dann weiß sie, daß es sich um das Steuerwort für die Betriebsart handelt.

Die Bits Nr. 4 und Nr. 5 haben *Don't Care*-Charakter (sprich: dount käär; nicht drum kümmern). Die Bits können beliebige Werte haben.

Entscheidend für die Betriebsart sind die Bits Nr. 6 und Nr. 7. Mit diesen beiden Bits wird der Z80 PIO binär verschlüsselt mitgeteilt, in welcher Betriebsart sie arbeiten soll.

Aufgabe S6.1

Am Anfang eines beliebigen Programms sind die beiden Befehle

```
LD    A,AFH
OUT   (PORT),A
```

programmiert. Mit dem zweiten Befehl wird das Byte aus dem Akkumulator als Befehls-Byte an die Z80 PIO geschickt.

Was bewirken diese beiden Befehle?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü1.

Die Programmierung der Betriebsart 3

Die zu Ihrem Lehrgang gehörende Interface-Leiterplatte haben wir zur vorzugsweisen Verwendung der Betriebsart 3 der Z 80 PIO ausgelegt. Wir haben bereits darauf hingewiesen, daß diese Betriebsart am universellsten verwendbar ist (vgl. Seite S 5). Zugunsten dieser Betriebsart wurde darauf verzichtet, auf der Leiterplatte die Anschlüsse READY und STROBE zugänglich zu machen.

Mit welchen Befehlen die Betriebsart 3 programmiert werden kann, wurde in den vorangegangenen Abschnitten gezeigt.

Aufgabe S 7.1

a) Geben Sie bitte die Befehlsfolge für die Programmierung des Ports B der Z 80 PIO im Micro-Professor-System zum Arbeiten in der Betriebsart 3 in mnemonischen Codes und mit den zugehörigen Bytes an, die Sie dazu über das Tastenfeld des Micro-Professors eintasten müssen!

b) Schreiben Sie die gleiche Befehlsfolge noch einmal an, verwenden Sie diesmal aber in beiden Befehlen Operanden, die von den zuerst angeschriebenen Operanden abweichen, mit denen aber ebenfalls die Programmierung für die Betriebsart 3 erreicht wird!

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 2.

In den Betriebsarten Ø und 1 ist durch das Betriebsart-Steuerwort (Mode-Wort) bereits vorgegeben, ob der entsprechende PIO-Port als Eingabe-Port (Empfänger) oder als Ausgabe-Port (Sender) arbeitet. In der Betriebsart 2 bestimmen die Handshake-Signale, ob der PIO-Port A jeweils als Sender oder als Empfänger arbeitet. Die Programmierung der PIO ist also eigentlich mit der Übertragung des Betriebsart-Steuerworts bereits abgeschlossen.

Wir werden Ihnen in einem der nächsten Abschnitte noch zeigen, daß bei den Betriebsarten Ø bis 2 nach dem Betriebsart-Steuerwort noch ein weiteres Befehlsbyte an die PIO geschickt werden muß. Zunächst soll der Hinweis genügen, daß z. B. in der Betriebsart 1 die Peripherie der PIO das Aussenden eines neuen Bytes mit einem STROBE-Ø-Signal meldet. Daraufhin schickt die PIO an die CPU ein Interrupt-Signal, mit dem das gerade laufende Programm unterbrochen und eine Interrupt-Service-Routine aufgerufen wird, die zum Einlesen des Bytes programmiert worden ist (vgl. Seite S 4). Damit das richtig funktioniert, muß die PIO der CPU mitteilen, wo die entsprechende Interrupt-Service-Routine abgelegt ist, und das wird mit einem zweiten Befehls-Byte an die PIO ermöglicht.

Machen Sie sich bitte über die (nur scheinbar!) komplizierten Zusammenhänge bei der Interrupt-Programmierung jetzt noch keine Gedanken. Wir wollten hier nur andeuten, was die Z 80 PIO so alles an Programmierung gebraucht. Sie sehen: In den Betriebsarten Ø bis 2 ist das nicht viel.

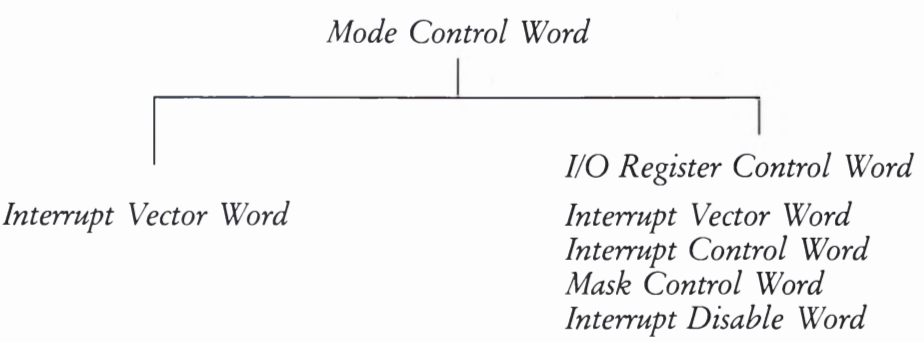
Etwas anders sieht das bei der Betriebsart 3 aus, die uns hier besonders interessieren soll.

S
8

Weil in dieser Betriebsart ganz nach Wunsch jedes der acht Bit eines PIO-Ports als Sender oder als Empfänger programmiert werden kann, muß dieses Sender-Empfänger-Bitmuster der PIO mit einem zweiten Befehls-Byte mitgeteilt werden, das als I/O-Register-Steuerwort *(I/O Register Control Word)* bezeichnet wird.

Auch in der Betriebsart 3 ist damit die Programmierung eigentlich bereits abgeschlossen. Auf den Seiten S5/ S6 haben wir jedoch angedeutet, daß es einige spezielle Möglichkeiten für die Interrupt-Programmierung gibt, die wir hier nur erwähnen, aber noch nicht erläutern wollen. Jedenfalls kann man für die Interrupt-Programmierung der Z 80 PIO in der Betriebsart 3 vier weitere Befehls-Bytes aussenden.

Insgesamt ergibt sich nach diesen Überlegungen für die Programmierung der Z 80 PIO folgendes Schema:



Wir haben hier ganz bewußt die englischen Bezeichnungen der Befehle angegeben, damit Sie sich ggf. in den (meist englisch abgefaßten) Datenblättern zurechtfinden. Wir haben das *Mode Control Word* als Betriebsart-Steuerwort bezeichnet. *I/O Register Control Word* läßt sich mit I/O-Register-Steuerwort übersetzen. (I/O ist die Abkürzung für *Input/Output:* Eingabe/Ausgabe.) – Die Steuerworte für die Interrupt-Funktionen erläutern wir in anderem Zusammenhang.

Das I/O-Register-Steuerwort

Bei der Programmierung eines Ports der Z 80 PIO für die Betriebsart 3 mit dem Betriebsart-Steuerwort CFH (Bitmuster 11XX 1111, vgl. Seite S6) ist es eine unbedingte Notwendigkeit, der PIO gleich anschließend mitzuteilen, welche der Port-Anschlüsse als Sender und welche als Empfänger arbeiten sollen. Diese Angabe macht ja gerade den Witz der Betriebsart 3 aus. Dem Betriebsart-Steuerwort muß also unbedingt sofort anschließend ein I/O-Register-Steuerwort folgen. So betrachtet ist der erste Befehl zur Programmierung der Betriebsart 3 ein zwei-Byte-Befehl. Solange man von den Interrupt-Möglichkeiten in dieser Betriebsart keinen Gebrauch macht, kann dann auf weitere Befehle verzichtet werden.

Das Bild S9.1 zeigt – entsprechend der Darstellung des Betriebsart-Steuerworts im Bild S6.1 – den Bit-weisen Aufbau des I/O-Register-Steuerworts. Jedem Bit innerhalb dieses Bytes ist ein entsprechend numerierter Anschluß eines PIO-Ports zugeordnet. Ein 0-Bit programmiert den zugeordneten Port-Anschluß als Ausgang (Sender, Output), ein 1-Bit programmiert den Anschluß als Eingang (Empfänger, Input).

Die Programmierung z. B. des Ports B der Z 80 PIO für die Betriebsart 3 ohne Berücksichtigung der Interrupt-Möglichkeiten bedarf folgender Befehle für die CPU:

```
LD    A,CFH        ;Betriebsart-Steuerwort in den Akkumulator
OUT   (83H),A      ;  und an Port-Adresse 83H ausgeben.
LD    A,I/OWord    ;I/O-Register-Steuerwort in d. Akkumulator
OUT   (83H),A      ;  und an Port-Adresse 83H ausgeben.
```

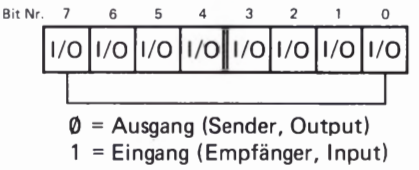


Bild S 9.1
Aufbau des I/O-Register-Steuerworts (I/O *Register Control Word*).

Aufgabe S 9.1

a) Geben Sie bitte die Befehlsfolge zur Programmierung des PIO-Ports A für die Betriebsart 3 an, wobei die Port-Anschlüsse A2, A4 und A7 als Eingänge, also als Empfänger für Signale von der Peripherie arbeiten. Die übrigen Port-Anschlüsse der PIO sollen als Ausgänge arbeiten, also Signale an die Peripherie aussenden. Die Arbeitsweise des Ports A sehen Sie im Bild S 9.2.

b) Schreiben Sie die Byte-Folge an, die Sie für die oben genannte Programmierung der Z 80 PIO als Programm über die Tastatur des Micro-Professors eingeben müssen.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 2.

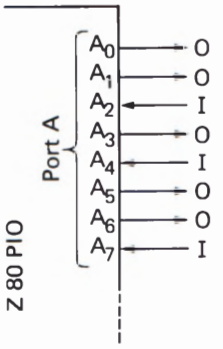


Bild S 9.2
Zu Aufgabe S 9.1: Die Anschlüsse des PIO-Ports A sollen so als Aus- und Eingänge arbeiten, wie es hier dargestellt ist.

Einlesen und Ausgeben von Daten über einen PIO-Port

Wir haben uns bisher fast ausschließlich damit beschäftigt, wie man die PIO zu einem bestimmten Verhalten programmieren kann. Von weitaus größerem Interesse ist doch eigentlich, wie denn nun der Datenverkehr zwischen der CPU und der Peripherie über den Schnittstellen-Baustein abgewickelt wird, wie also die CPU die bei der PIO abgelieferten Bytes abholt oder wie die CPU der PIO Bytes zum übertragen an die Peripherie übergibt.

Sie wissen bereits, daß für den Datenverkehr zwischen der CPU und der PIO die Datenport-Adressen der PIO zuständig sind. Die Sache ist also einfach. Vor dem Beginn des Datenverkehrs zwischen CPU und Peripherie auf dem Weg über die PIO werden von der CPU an die Befehls-Portadressen der PIO einige Befehls-Bytes zum Programmieren des Bausteine geliefert. Wenn das geschehen ist, dann treten nur noch die Daten-Portadressen in Funktion.

Daten zur Übermittlung an die Peripherie liefert die CPU bei der PIO nach dem gleichen Schema ab, nach dem auch Befehls-Bytes abgeliefert werden:

```
LD    A,Byte           ;Zu übertragendes Byte in den Akkumulator
OUT   (Datenport),A    ;  und bei der Datenport-Adresse abliefern.
```

Ganz entsprechend holt sich die CPU auch Daten von der PIO ab:

```
IN    A,(Datenport)    ;Byte von der Datenport-Adresse zum Akku.
```

Ähnlich wie bei der Programmierung der PIO kann es sich bei diesen Befehlen um den Datenport für den PIO-Port A oder um den für den PIO-Port B handeln.

S

10

Versuche auf der Peripherie-Leiterplatte

Sie haben sich bisher mit der Handhabung der Z80 PIO nur recht theoretisch beschäftigen können. Das war kaum anders möglich, denn Sie haben gesehen, daß der Schnittstellen-Baustein mit den Eigenschaften einer CPU ein nicht ganz unkompliziertes Gebilde ist, dessen Fähigkeiten man erst kennen muß, ehe man sie sinnvoll einsetzen kann.

Wir wollen Ihnen die Möglichkeiten, die der Einsatz der Z80 PIO bietet, in einigen Programmen vorstellen. Es sei ausdrücklich darauf hingewiesen, daß es weniger auf die Programme selbst ankommt, die keine speziellen Nutzanwendungen bieten. Wir wollen einfach eine Aufgabe stellen und dann zusehen, wie sich die Aufgabe unter Verwendung der Eigenschaften der Z80 PIO lösen läßt.

Von Hand gesteuerte Daten-Ausgabe

Die erste dieser Aufgaben sei einfach die, einer Peripherie Daten über acht Datenleitungen zu schicken. Die Signalwerte auf jeder dieser acht Datenleitungen sollen unabhängig voneinander über jeweils einen Tastschalter vom Signalwert Ø auf den Signalwert 1 und umgekehrt wieder mit dem gleichen Tastschalter vom Signalwert 1 auf den Signalwert Ø umgeschaltet werden können.

Das System soll so arbeiten, als enthielte es acht D-Flipflops, deren Ausgangssignale über jeweils einen Schalter abwechselnd zwischen den Werten Ø und 1 hin- und hergeschaltet werden können.

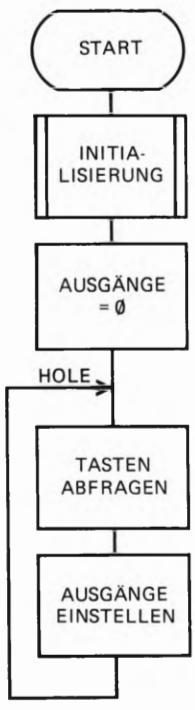


Bild S10.1
Ablaufplan des Programms für die von Hand gesteuerte Daten-Ausgabe.

Mit dieser Aufgabenstellung wird verlangt, daß das System acht voneinander unabhängige 1-Bit-Eingänge hat und ebensoviele 1-Bit-Ausgänge, die von den Eingängen gesteuert werden. Die Aufgabe läßt sich von der in der Betriebsart 3 arbeitenden Z80 PIO lösen.

Vorab wollen wir noch überlegen, ob diese Aufgabe auch ohne eine spezielle Schnittstellen-Anordnung mit dem Micro-Professor lösbar wäre.

Grundsätzlich könnte man als Eingabe-Tasten natürlich 8 beliebige Tasten des Micro-Professor-Tastenfelds verwenden, wenn man diese Tasten für die gestellte Aufgabe per Software entsprechend decodiert. Mit jeder dieser Tasten könnte z.B. ein spezielles Bit in einem der CPU-Register umgeschaltet werden.

Problematisch wird die Sache allerdings, wenn es um die Ausgabe der eingegebenen Daten geht. Man kann zwar den Inhalt eines beliebigen Registers (ggf. auf dem Weg über den Akkumulator) mit einem OUT-Befehl auf den Datenbus schalten und gleichzeitig über die niederwertigen acht Leitungen des Adreßbus bestimmen, wohin dieser Inhalt geschickt werden soll. Das System setzt dann allerdings voraus, daß der Register-Inhalt bei der Zieladresse aufbewahrt wird, denn das ist auf dem Datenbus nicht möglich. Gleich bei der Ausführung des nächsten Befehls wird die CPU den Datenbus wahrscheinlich für ganz andere Daten benötigen und die vorherigen Daten wieder löschen.

Die über den Datenbus ausgesandten acht Bits aus dem Register müßten in acht Flipflops aufbewahrt und an deren Ausgängen der Peripherie zur Verfügung gestellt werden. Die Flipflops müßten vom OUT-Befehl adressiert und dann über den Datenbus auf neue Inhalte eingestellt werden.

Mit dieser Überlegung landen wir automatisch bei der Funktion eines Schnittstellen-Bausteins, denn im Prinzip enthält dieser Baustein an den Leitungen eines Ausgabe-Ports auch nichts anderes als die hier beschriebenen Flipflops. Nur ist der Schnittstellen-Baustein etwas komfortabler eingerichtet. Die Z 80 PIO enthält z. B. nicht nur einmal, sondern zweimal acht Ausgabe-Flipflops. Außerdem können die Ausgabe-Leitungen auch als Eingabe-Leitungen programmiert werden usw. Aber diese Möglichkeiten haben wir Ihnen ja bereits vorgestellt.

Kehren wir zurück zu der gestellten Aufgabe. Den grundsätzlichen Ablauf des (recht kurzen) Programms, das diese Aufgabe mit Hilfe der Z 80 PIO lösen kann, haben wir im Bild S 10.1 dargestellt.

Es wurde bereits auf der Seite S 1 erwähnt, daß die zum Lösen einer Aufgabe notwendige Einstellung der Ports des Schnittstellen-Bausteins vor den Befehlen programmiert werden muß, mit denen das System die Aufgabe selbst löst. Wir bezeichnen diese Einstellung der PIO-Ports als Initialisierung. Die dafür notwendigen Befehle werden in einem Unterprogramm zusammengefaßt, das wir nachher vorstellen. Das Unterprogramm wird gleich am Anfang des Programms aufgerufen.

In unserem Programm wird der PIO-Port A als Eingabe-Port verwendet. Über diesen Port werden die von den zugehörigen Tastschaltern gelieferten Signale (vgl. die Bilder H 12.1 und H 14.1) eingelesen.

Über den PIO-Port B werden die Signale an die Peripherie ausgegeben und von den entsprechenden Leuchtdioden angezeigt. Im Bild H 12.1 erkennen Sie, daß die über den PIO-Port B ausgegebenen Signale auf der Peripherie-Leiterplatte bei Bedarf tatsächlich für eine echte Peripherie verfügbar sind: Sie sind an die Anschlüsse 0 bis 7 der Klemmleiste geführt (vgl. Bild H 9.1).

Gleich anschließend an die Initialisierung schickt die CPU das Byte ØØ zur Ausgabe über den Port B an die PIO, so daß zu Beginn des Programms über den Port B nur Ø-Signale ausgegeben werden und die Ausgabe-Leuchtdioden dunkel sind.

Das Hauptprogramm beginnt beim Label HOLE und läuft dann in einer Schleife. Mit der ersten Anweisung werden die Eingabe-Tasten abgefragt. Dieser Programmteil wird erst dann verlassen, wenn eine Taste betätigt worden ist. Im Bild S 11.1 ist dieser Programmteil detailliert dargestellt.

In der zweiten Schleifen-Anweisung veranlaßt das zur betätigten Taste gehörende 1-Bit über einen XOR-Befehl (vgl. Lehrgang Mikroprozessortechnik, Seiten S 30 und S 56) das Umschalten des zur Taste gehörenden Ausgangssignals vom Wert Ø auf den Wert 1 bzw. vom Wert 1 zurück auf den Wert Ø.

Ehe wir Ihnen die besonders interessierenden Teile des Programms genauer vorstellen, sollen Sie sich ansehen, wie das Programm funktioniert.

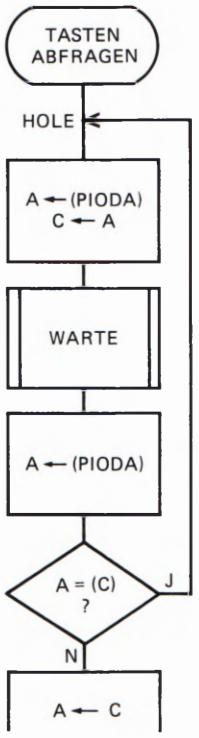


Bild S 11.1
Unterprogramm für das TASTEN-ABFRAGEN im Programm für die von Hand gesteuerte Daten-Ausgabe.

S
12

Versuch S 12.1

Ausgabe-Flipflops

Verbinden Sie die Peripherie-Leiterplatte über die beiden Flachbandleitungen mit Ihrem System (vgl. Seite H10) und schließen Sie die Stromversorgung an.

Damit die zum PIO-Port A gehörenden Tastschalter Eingangssignale für diesen Port generieren können, müssen die acht steckbaren Brükken für die Schalter des Ports A auf die entsprechenden Stiftleisten auf der Peripherie-Leiterplatte gesteckt werden (vgl. Bild H14.1). Sie finden die steckbaren Brücken in hinreicher Anzahl in ihrer Materialsendung.

Tasten Sie die zum Programm auf den Seiten L11 und L12 gehörenden Bytes ab der Adresse 1800H in Ihr System ein. (Lesen Sie bitte noch einmal die Hinweise auf den Seiten L1 und L2 durch!)

Überzeugen Sie sich, daß Sie alle Bytes fehlerfrei eingegeben haben, und starten Sie dann das Programm bei der Adresse 1800H! – Die Anzeige auf dem Micro-Professor verlischt, und auch die Leuchtdioden auf der Peripherie-Leiterplatte sind zunächst dunkel.

Betätigen Sie in beliebiger Reihenfolge die zum Port A gehörenden Tastschalter auf der Peripherie-Leiterplatte und beobachten Sie die Leuchtdioden!

Beim Betätigen eines Tastschalters leuchtet die zugehörige Leuchtdiode des Ports A auf. Gleichzeitig ändert sich der von der darüberliegenden Leuchtdiode angezeigte Wert des Ausgangssignals im Port B. Nur dieser eine Signalwert wird jeweils geändert; alle übrigen Signalwerte bleiben unverändert. Dieses Verhalten entspricht genau unserer Aufgabenstellung.

Sehen Sie sich das Unterprogramm INIT zur Initialisierung der PIO auf der Seite L12 an! Im Versuch werden beide PIO-Ports (A und B) verwendet. Dementsprechend müssen auch beide Ports eigens programmiert werden.

Das Prinzip der Programmierung haben wir Ihnen am Beispiel des Ports B auf der Seite S9 vorgestellt. Sie haben dort gesehen, daß zur Programmierung der Betriebsart 3 eigentlich vier Befehle notwendig sind. In unserem Programm kommen wir offenbar bei der Programmierung mit jeweils drei Befehlen aus.

Der Grund ist einfach: Wir nutzen die *Don't-Care*-Werte im Steuerwort für die Betriebsart 3 (vgl. das Bild S6.1) aus.

Der PIO-Port A muß auf sämtlichen acht Leitungen als Empfänger für die Tasten-Eingaben programmiert werden. Das I/O-Register-Steuerwort für diesen Port muß also das Bitmuster 1111 1111 (= FFH) haben (Bild S9.1). Dieses Byte muß als zweites Befehls-Byte an die Befehls-Adresse des PIO-Ports A geschickt werden.

Das Steuerwort für die Betriebsart 3, das als erstes Befehls-Byte an die Befehls-Adresse des PIO-Ports A geschickt wird, hat das Bitmuster 11XX 1111. Nichts hindert uns daran, für die mit X gekennzeichneten *Don't-Care*-Bits die Werte 1 einzusetzen. Dann ergibt sich für die Übermittlung des Betriebsart-Steuerworts die Befehlsfolge

```
LD   A,11111111B
OUT  (PIOBA),A
```

Bei der mnemonischen Formulierung des ersten der beiden Befehle kennzeichnen wir den Operanden 11111111 durch den nachgestellten Buchstaben B als Binär-Darstellung. – PIOBA im zweiten Befehl bezeichnet die Port-Adresse 82H für **PIO-B**efehle für den Port **A**.

Anschließend an diese beiden Befehle muß das I/O-Register-Steuerwort an die Adresse 82H der PIO geliefert werden. Das würde normalerweise mit der Befehlsfolge

```
LD   A,11111111B
OUT  (PIOBA),A
```

geschehen. Da aber gerade vorher bereits das Byte 1111 1111B in den Akkumulator geladen wurde und sich dieser Wert durch den OUT-Befehl nicht geändert hat, können wir uns die erneute Programmierung des LD A,11111111B-Befehls sparen.

Jetzt muß der PIO-Port B für die Betriebsart 3 programmiert werden. Als erstes Befehls-Byte muß die Bitfolge 11XX 1111 an die Port-Adresse 83H für **PIO-B**efehle für den Port **B** (PIOBB) geschickt werden. Auch hier setzen wir für die *Don't-Care*-Bits X die Werte 1 ein. Es ergibt sich das Befehls-Byte 1111 1111B, das von der Programmierung des Ports A her bereits im Akkumulator steht. Die Programmierung des Befehls LD A,11111111B erübrigt sich also, und mit dem Befehl OUT (PIOBB),A kann das Betriebsart-Steuerwort sofort ausgegeben werden.

Die acht Datenleitungen des PIO-Ports B müssen als Sender für die Signal-Ausgabe an die Peripherie programmiert werden. Das I/O-Register-Steuerwort hat entsprechend das Bitmuster ØØØØ ØØØØ. Dieses Byte steht nicht im Akkumulator. Vor dem OUT-Befehl muß der Akkumulator-Inhalt gelöscht werden. Wir haben dazu den Befehl LD A,ØØ programmiert. (Noch sparsamer, nämlich mit einem Ein-Byte-Befehl, hätten wir das mit dem Befehl XOR A erreicht. (Vgl. Lehrgang Mikroprozessortechnik, Seite S 59.) – Als letzter Befehl der Initialisierungs-Routine kann nun der Befehl OUT (PIOBB),A folgen.

Die restlichen Anweisungen des Programms bedürfen nur weniger Erläuterungen. – Da sich beim Ablauf unseres Programms an der Arbeitsweise der PIO-Ports nichts mehr ändert, brauchen auch keine weiteren Befehls-Bytes an die Befehls-Portadressen 82H (PIOBA) und 83H (PIOBB) geschickt zu werden. Zwischen der PIO und der CPU findet jetzt nur noch reine Datenbyte-Kommunikation statt.

Wir wollen bei dieser Gelegenheit darauf hinweisen, daß es innerhalb eines Programms durchaus möglich ist, die Programmierung der PIO-Ports zu ändern, wenn sich das als notwendig erweist. In einem solchen Fall braucht die CPU nur entsprechende Befehls-Bytes an die Befehlsadresse des Ports zu senden, dessen Eigenschaften geändert werden sollen.

In unserem Programm folgt nach der Initialisierung der PIO die Anweisung, auf die Ausgangsleitungen zunächst die Signalwerte Ø zu schalten (Leuchtdioden aus). Dazu wird in den Akkumulator das Byte ØØØØ ØØØØB geladen und dieses Byte dann an die Port-Adresse 81H (PIODB) geschickt, die für die **PIO-D**aten für den Port **B** zuständig ist.

S
14

Das Laden des Akkumulators mit dem Byte ØØ hätten wir uns eigentlich sparen können, denn dieses Byte steht sowieso im Akkumulator, wenn die INIT-Routine erledigt ist. Da man diese Tatsache jedoch der vorhergehenden Anweisung CALL INIT nicht ansehen kann, ist es eine gute Methode, solche an sich überflüssige Anweisungen im Sinne guter Lesbarkeit von Programmen zu programmieren.

In der Schleife des anschließenden Hauptprogramms wird zunächst zweimal in kurzen Abständen nacheinander mit IN-Befehlen abgefragt, ob über die für PIO-Daten für den Port A zuständige Adresse 8ØH (PIODA) mit einem 1-Bit die Betätigung einer Taste gemeldet wird. Erst wenn das der Fall ist, wird im H-Register eine Änderung des entsprechenden Bits vorgenommen. (Wir verwenden den Inhalt des H-Registers in unserem Programm als Bild der über den PIO-Port B an die Peripherie gelieferten Daten.) Anschließend wird das über die betätigte Taste geänderte Bitmuster über die Portadresse 81H (PIODB) mit einem OUT-Befehl ausgegeben. Die Portadresse 81H ist für PIO-Daten für den Port B zuständig.

Bitte beachten Sie, daß wir die Bezeichnungen PIODA, PIODB usw. für unser Programm willkürlich gewählt haben. Selbstverständlich kann man auch andere Bezeichnungen für die Portadressen wählen.

Gesteuertes Lauflicht

Bild S14.1
An den hier rot markierten Stellen müssen für das gesteuerte Lauflicht auf den Stiftleisten für steckbare Brücken Kurzschlußstecker angebracht werden.

Das im folgenden vorgestellte Lauflicht-Programm bietet – abgesehen davon, daß es recht hübsch ist – keinerlei Nutzanwendung. Es ist aber gut dafür geeignet, die Fähigkeiten der Z 80 PIO zu zeigen und ist in seiner Auslegung um einiges anspruchsvoller als das Programm aus dem vorhergehenden Abschnitt.

Das Bild S14.2 deutet an, welche Aufgabe dem Programm zugrunde liegt. Die zu den Bits Nr. 6 bis Nr. 1 der Ports A und B gehörenden Leuchtdioden sollen so gesteuert werden, daß ein Lichtpunkt im Kreis herum läuft. Das Umlaufen kann im Uhrzeigersinn und entgegengesetzt erfolgen; es kann auch unterbrochen und wieder gestartet werden.

Die Steuerung des Umlaufens wird über die zu den Bits Nr. 7 und Nr. 0 der beiden Ports gehörenden Tasten vorgenommen. Welche Funktionen diesen Tasten zugeordnet werden, ist im Bild S14.2 ebenfalls angedeutet.

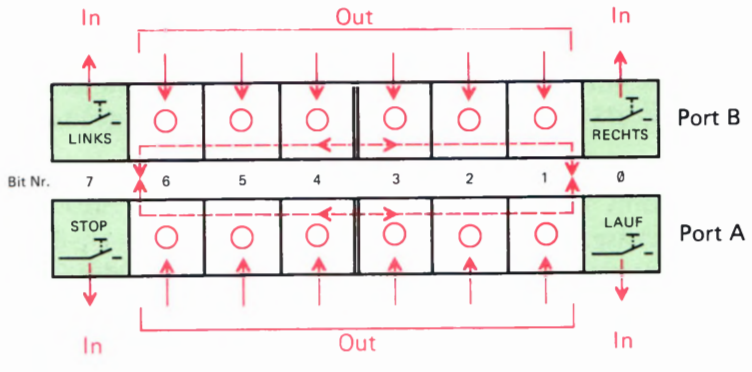


Bild S14.2
Das gesteuerte Lauflicht benutzt je 6 Anschlüsse der beiden Ports. Je zwei Port-Anschlüsse werden für die Eingabe von Steuersignalen verwendet.

Die Aufgabenstellung zeigt bereits, wie die beiden Ports A und B programmiert werden müssen: Bei beiden Ports müssen die Anschlüsse Nr. 6 bis Nr. 1 als Daten-Sender arbeiten; die Anschlüsse Nr. 7 und Nr. 0 werden als Daten-Empfänger programmiert (Bild S 15.1).

Ehe wir die in diesem Zusammenhang besonders interessierenden Teile des Programms erläutern, sollen Sie sich wieder ansehen, wie die Sache in Ihrem System funktioniert.

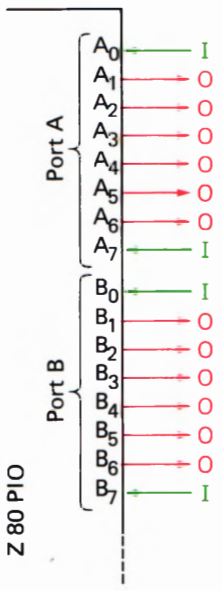


Bild S 15.1
So werden die Port-Anschlüsse für das gesteuerte Lauflicht als Ein-und Ausgänge programmiert.

Versuch S 15.1

Gesteuertes Lauflicht

Schließen Sie die Peripherie-Leiterplatte an Ihr System an, und vergessen Sie die Stromversorgung nicht.

Das Bild S 14.1 zeigt, wie Sie die Stiftleisten für die steckbaren Brücken mit Kurzschlußsteckern bestücken müssen, damit mit den Tastschaltern Nr. 7 und Nr. 0 der beiden Ports Steuersignale eingegeben werden können. (Vgl. das Bild H 12.1.) *)

Tasten Sie die zum Lauflicht-Programm ab der Seite L 13 gehörenden Bytes ab der Adresse 1800H in Ihr System ein. Wenn Sie sich davon überzeugt haben, daß Ihre Eingaben fehlerfrei sind, können Sie das Programm bei der Adresse 1800H starten. Die Anzeige des Micro-Professors wird verlöschen.

Das Programm ist so ausgelegt, daß nach seinem Start zunächst kein Lichtpunkt umläuft. – Betätigen Sie auf der Peripherie-Leiterplatte die zum Port A gehörende Taste mit der Bezeichnung 0! Im Bild S 14.2 sehen Sie, daß das zu dieser Taste gehörende 1-Bit mit einem IN-Befehl eingelesen wird. Das Programm interpretiert das 1-Bit als Kommando „Lauf"; der Lichtpunkt beginnt ab dem Bit Nr. 6 des PIO-Ports B im Uhrzeigersinn (also rechts herum) umzulaufen.

Betätigen Sie die Taste 7 des Ports B! Die Laufrichtung des Lichtpunkts kehrt sich um. Sie kann mit der Taste 0 des Ports B wieder in den Uhrzeigersinn gesteuert werden.

Betätigen Sie die zum Port A gehörende Taste 7! Der Lichtpunkt bleibt stehen. Mit der Taste 0 des Ports A kann er von dieser Stellung aus zu weiteren Umlaufen wieder gestartet werden.

Versuchen Sie, den Umlaufsinn bei angehaltenem Umlauf umzusteuern und starten Sie dann den Umlauf wieder. Auch das wird gelingen.

Dies ist ein hübsches Spielchen. Viel wichtiger ist es aber, wie das Programm diese Steuerung des Lichtpunkts erreicht. – Wir wollen das Programm nicht in allen Einzelheiten erläutern. Wenn Sie das interessiert, dann geben Ihnen die Kommentare in der Programm-Auflistung die notwendige Auskunft. Hier soll nur das Prinzip der Steuerung gezeigt werden. Vor allem wollen wir aufzeigen, welche Rolle die Programmierung der Z 80 PIO in diesem Programm spielt.

Im Bild S 16.1 sehen Sie den grundsätzlichen Ablaufplan des Programms. Es ist Ihnen bereits geläufig, daß gleich am Anfang des Programms die Befehle zur Initialisierung des Schnittstellen-Bausteins stehen müssen. Die zugehörigen Anweisungen sind wieder als Unterprogramm INIT programmiert, damit das Programm selbst übersichtlich bleibt. Wir stellen die INIT-Routine nachher genauer vor.

*) die entsprechenden Dual im Dual Schalter sind auf OFF zu setzen

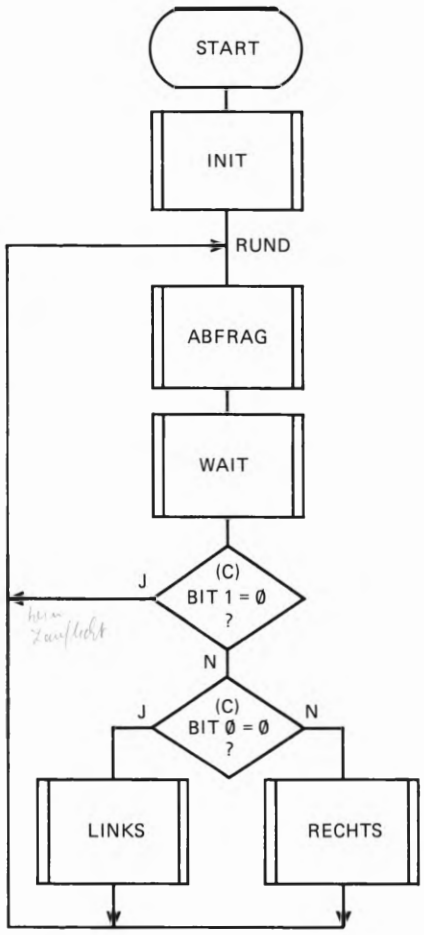


Bild S16.1
Programmablaufplan für das gesteuerte Lauflicht.

Nach der INIT-Routine läuft das Programm ab dem Label RUND in einer Schleife. In der ABFRAG-Routine wird nachgesehen, ob eine der vier Steuertasten (jeweils die Tasten 0 und 7 der Ports A und B) betätigt wurde. Wenn das nicht der Fall ist, dann wird sofort anschließend die Warte-Schleife WAIT abgearbeitet, die dafür sorgt, daß der Lichtpunkt langsam umläuft. Ohne diese WAIT-Schleife liefe der Lichtpunkt so schnell um, daß das Auge den Eindruck hat, als leuchteten sämtliche Lichtpunkte gleichzeitig. Sie können sich das ansehen, wenn Sie versuchsweise die drei Bytes für den Befehl CALL WAIT ab der Adresse 18ØCH durch NOP-Befehle (Bytes ØØ) ersetzen.

Wird in der ABFRAG-Routine die Betätigung einer der vier Steuertasten festgestellt, dann sorgen die Anweisungen in dieser Routine dafür, daß im C-Register entsprechend der betätigten Taste die Bits Nr. 0 und Nr. 1 für den gewünschten weiteren Ablauf der Schleife eingestellt werden. Das Bild S16.2 zeigt die Bedeutung der Bits Nr. 0 und Nr. 1 im C-Register, die den Charakter von Flags haben. – Mit dem Befehl LD C,ØØØØØØØ1B (LD C,Ø1) bei der Adresse 18Ø7H wird der Inhalt des C-Registers vor dem Eintritt in die Schleife des Hauptprogramms so eingestellt, daß nach dem Start des Programms der Lichtpunkt gestoppt bleibt und der Umlauf im Uhrzeigersinn vorprogrammiert ist.

Nach dem Abarbeiten der WAIT-Schleife sieht sich das Hauptprogramm zunächst das Bit Nr. 1 im C-Register an. Wenn dieses Bit den Wert 0 hat, dann erfolgt sofort ein Rücksprung zum Label RUND; der Lichtpunkt wird nicht von der Stelle bewegt.

Hat das Bit Nr. 1 im C-Register nicht den Wert 0 (also den Wert 1), dann muß die Entscheidung getroffen werden, ob der Lichtpunkt im Uhrzeigersinn (RECHTS) oder entgegengesetzt (LINKS) bewegt werden soll. Diese Entscheidung ist vom Wert des Bits Nr. 0 im C-Register abhängig, und entsprechend wird die Routine RECHTS oder LINKS aufgerufen.

Wie diese Routinen im einzelnen arbeiten, können Sie sich in der Auflistung des Programms ansehen. Hier werde nur erwähnt, daß der Lichtpunkt mit einem 1-Bit im H-Register abgebildet wird, wenn er über den Port A ausgegeben werden soll. Er wird mit einem 1-Bit im L-Register abgebildet, wenn er über den Port B ausgegeben werden soll. Sie finden in den LINKS- und RECHTS-Routinen jeweils einige Befehle OUT (PIODA),A und OUT (PIODB),A, die diese Ausgaben besorgen. Wenn Sie diese Befehle in der Programm-Auflistung aufsuchen, dann finden Sie in den meisten Fällen unmittelbar davor einen Befehl, der das Bitmuster aus dem H- oder L-Register zur Ausgabe in den Akkumulator bringt. (Ausnahme: Der Leuchtpunkt soll ganz gelöscht werden. In diesem Fall wird der Akkumulator-Inhalt ØØ mit einem OUT-Befehl ausgegeben. Der Befehl XOR A sorgt dafür, daß im Akkumulator das Byte ØØ steht.)

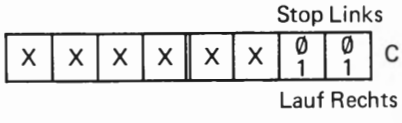


Bild S16.2
Die Bits Nr. 0 und Nr. 1 des C-Registers dienen als Flags für die Laufrichtung und Lauf/Stop.

Sehen Sie sich jetzt bitte das Unterprogramm INIT zur Initialisierung der Z80 PIO in der Programm-Auflistung an. Die zur Programmierung der beiden PIO-Ports A und B eingesetzten Befehlsfolgen entsprechen in diesem Programm – anders als im Programm zum vorhergehenden Versuch – genau der Programmier-Befehlsfolge, die wir auf der Seite S9 vorgestellt haben. Irgendwelche Tricks zur Verkürzung der Befehlsfolgen lassen sich hier nicht verwenden.

Das erste Befehls-Byte ist jeweils das Steuerwort für die Betriebsart 3. Wir verwenden das Byte FFH mit dem Bitmuster 1111 1111, haben also den beiden *Don't-Care*-Bits Nr. 5 und Nr. 4 (vgl. Seite S6) die Werte 1 zugeordnet. Mit dem gleichen Erfolg hätten wir auch als Betriebsart-Steuerwort das Byte CFH mit dem Bitmuster 11ØØ 1111 oder das Byte EFH mit dem Bitmuster 111Ø 1111 programmieren können. Machen Sie einen Versuch!

Das zweite Befehls-Byte, das sowohl an den Befehls-Port PIOBA (Adresse 82H) als auch an den Befehls-Port PIOBB (Adresse 83H) der PIO übermittelt wird, ist das I/O-Register-Steuerwort. Im Bild S15.1 erkennen Sie, daß die Befehls-Bytes für beide Ports die gleiche Struktur haben. In beiden Fällen werden die Port-Anschlüsse Nr. Ø und Nr. 7 als Eingänge, also als Empfänger für Tasten-Steuersignale programmiert und die Port-Anschlüsse Nr. 1 bis Nr. 6 als Ausgänge, also als Sender für Leuchtpunkt-Daten.

Interessant ist der Aufbau der ABFRAG-Routine, deren Ablauf im Bild S17 dargestellt ist. Die Steuertasten an den Anschlüssen Nr. 0 und Nr. 7 werden zuerst beim PIO-Daten-Port B und dann beim PIO-Daten-Port A abgefragt. Die Programm-Auflistung zeigt die Einzelheiten.

Bei der Adresse 183CH wird das am Port B stehende Bitmuster mit einem IN-Befehl in den Akkumulator geholt. Dann werden die interessierenden Bits Nr. 7 und Nr. 0 ausmaskiert und festgestellt, ob über eine der Tasten ein 1-Bit zur Änderung der Laufrichtung eingegeben wurde. Ist das nicht der Fall, dann werden sofort auf die gleiche Weise ab dem Label LAUSTO die am Port A liegenden Tasten für die Kommandos 'Lauf' und 'Stop' abgefragt. Wenn auch diese Abfrage negativ ausfällt, dann wird die Routine verlassen, ohne daß am Inhalt des C-Registers etwas geändert wird.

Wenn bei einer der Abfragen festgestellt wird, daß eine Steuer-Taste betätigt wurde, dann wird untersucht, ob mit dieser Taste auf den Port-Anschluß Nr. 0 oder auf den Port-Anschluß Nr. 1 ein 1-Bit gesetzt wurde. Bei der Abfrage des Ports B wird – je nach betätigter Taste – der Wert 0 oder der Wert 1 so als Bit Nr. 0 in das C-Register eingetragen, daß der Wert des Bits Nr. 1 unverändert bleibt.

Wenn bei der Abfrage des Ports A festgestellt wird, daß eine der an diesen Port angeschlossenen Steuer-Tasten betätigt wurde, dann wird auch hier – je nach betätigter Taste – der Wert 0 oder der Wert 1 im C-Register abgelegt. Jetzt erfolgt die Eintragung jedoch an der Stelle des Bits Nr. 1, und zwar wiederum so, daß der Wert des Bits Nr. 0 unverändert bleibt.

Die WAIT-Routine, durch welche die Umlauf-Geschwindigkeit des Lichtpunkts bestimmt wird, ist etwas ungewöhnlich aufgebaut. Normalerweise wird für eine solche Routine ein 8-Bit-Register mit einem bestimmten Wert geladen und dieses Register dann in einer Schleife leergezählt: Solange im Register noch nicht der Wert ØØ steht, wird die Schleife, in der ein Dekrementier-Befehl steht, von neuem durchlaufen. Je höher der Wert ist, mit dem das Register anfangs geladen war, um so öfter muß die Schleife durchlaufen werden, um so länger ist die erreichte Verzögerung.

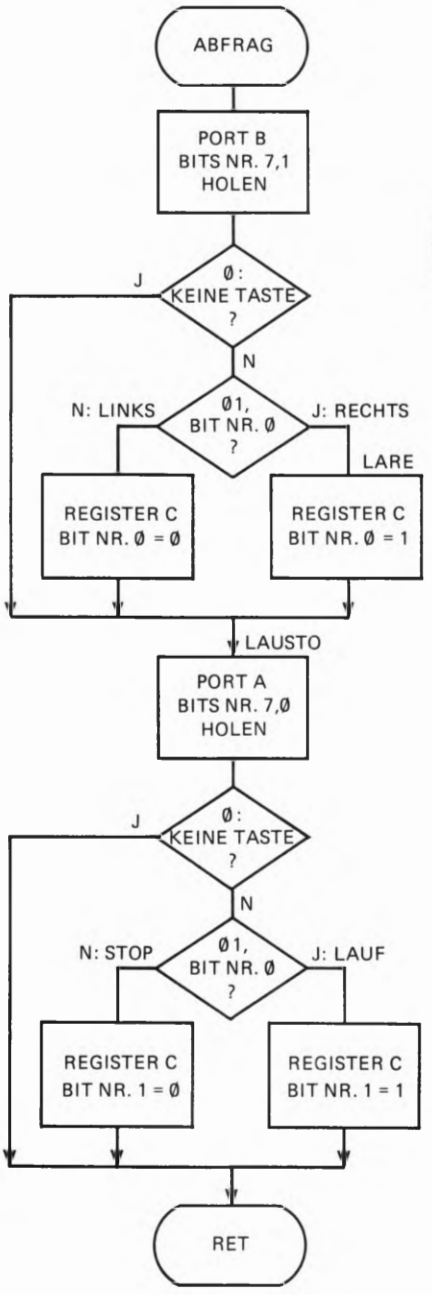


Bild S17.1
Ablaufplan der ABFRAG-Routine für die Steuertasten beim Lauflicht-Programm.

S
18

Wir verwenden in unserem Programm das 16 Bit breite HL-Registerpaar als Zähl-Register, das anfangs mit einem bestimmten Wert geladen wird. In der Schleife wird der Inhalt des Registerpaares nicht dekrementiert, sondern jeweils um den Inhalt des DE-Registerpaares erhöht. Wir setzen (DE) = 1, so daß (HL) jeweils um eins inkrementiert wird, bis der Wert (1)ØØØØ ØØØØ erreicht ist. Die hier in Klammern gesetzte Ziffer 1 hat im HL-Registerpaar keinen Platz mehr; sie erscheint als Carry-Bit im Flag-Register. (Vgl. Lehrgang Mikroprozessortechnik, Seite S 83.) Abhängig davon, ob das Carry-Bit bereits gesetzt ist oder noch nicht, wird die Bearbeitung der Inkrementierungs-Schleife abgebrochen oder fortgesetzt.

Im Gegensatz zur normalen Verzögerungs-Routine ist jetzt die Verzögerung um so kürzer, je größer der Wert ist, mit dem das HL-Registerpaar anfangs geladen wird: Um so schneller wird der Wert (1)ØØØØ ØØØØ erreicht. – Mit der hier gewählten Art der Programmierung können verhältnismäßig lange Verzögerungen erreicht werden.

Ändern Sie versuchsweise in Ihrem Programm den Operanden des Befehls LD (HL),ØE8ØØH bei den Adressen 1832H und 1833H und beobachten Sie dann die Umlaufgeschwindigkeit des Lichtpunkts!

Aufgabe S 18.1

Überlegen Sie bitte, warum bei dem im Versuch S 15.1 beschriebenen Lauflicht-Programm bei jedem Umlauf des Lichtpunkts eins der Relais auf der Peripherie-Leiterplatte deutlich „Klick" sagt.

Welches der beiden Relais ist das?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 2.

Die Interrupt-Struktur des Z 80

Im Laufe unseres Lehrgangs haben wir nun schon so oft von Interrupt geschrieben (Seiten H 8, S 4, S 7 und in der Einleitung zu diesem Lehrbrief), daß Sie sicher recht neugierig sind, was es denn damit eigentlich auf sich hat. Wir wollen Ihnen auf den folgenden Seiten zeigen, wie ein Interrupt funktioniert, was man damit anfangen kann und vor allem, wie man die Möglichkeiten des Interrupts in ein Programm einbauen kann.

Sie werden sehen, daß die zur Z 80-Familie gehörenden Peripherie-Bausteine ihr volles Können erst dann entfalten, wenn man von den Möglichkeiten des Interrupts Gebrauch macht.

Ein einfaches Programm mit Interrupt

Ehe wir Ihnen erläutern, wie ein Interrupt funktioniert, sollen Sie sich an einem sehr einfachen Programm selbst ansehen, wozu ein Interrupt überhaupt nützlich ist. Anhand dieses Programms erkennen Sie am besten, wie die Z 80-CPU auf einen Interrupt reagiert.

Versuch S 19.1

Unterbrechung eines Lauflichts

Verbinden Sie die Peripherie-Leiterplatte mit Ihrem System (vgl. Seite H 10) und schließen Sie die Stromversorgung an.

Tasten Sie bitte die zum Programm auf den Seite L 23 bis L 26 gehörenden Bytes ab der Adresse 1800H in Ihr System ein und vergessen Sie nicht, auch die beiden letzten Bytes ab der Adresse 1858H einzugeben!

Bestücken Sie die steckbaren Brücken der Stiftleisten zum Port A so, wie es das Bild S 19.1 zeigt. Mit den beiden Brücken werden die Tasten Nr. 0 und Nr. 1 der unteren Tastenreihe aktiviert.

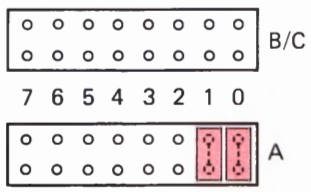


Bild S 19.1
Bestücken Sie für den Versuch S 19.1 die steckbaren Brücken für den Port A so, wie es hier rot eingetragen ist.

Starten Sie das Programm bei der Adresse 1800H! – Auf der Peripherie-Leiterplatte läuft in der zum PIO-Port B gehörenden Leuchtdioden-Zeile von rechts nach links ein Lauflicht.

Ein Programm für ein solches Lauflicht ist nun absolut nichts Besonderes; es kann mit ganz wenigen Befehlen programmiert werden.

Betätigen Sie kurzzeitig die zum PIO-Port A gehörende Taste Nr. 0 auf der Peripherie-Leiterplatte! Was jetzt geschieht, ist nicht zu überhören. Betätigen Sie die Taste noch einige Male! Was fällt Ihnen auf?

Sehen Sie sich bitte den Teil des soeben eingegebenen Programms an, der für das Laufen des Lichts zuständig ist. Der Initialisierungs-Befehl beim Label START ist in diesem Zusammenhang zunächst uninteressant. Mit dem zweiten Befehl wird das Bitmuster 00000001 in den Akkumulator geladen. Anschließend wird dieses Bitmuster beim Label LOOP mit dem OUT (PIODB),B-Befehl über den PIO-Port B ausgegeben; das Bit Nr. 0 mit dem Wert 1 in diesem Bitmuster schaltet

die zugehörige Leuchtdiode an. (In der INIT-Routine ist der Port B in der Betriebsart 3 als Ausgabe-Port programmiert worden.)

Der folgende RLCA-Befehl rotiert das Bitmuster im Akkumulator um ein Bit links herum, so daß jetzt das Bit Nr. 1 den Wert 1 hat. (Vgl. Lehrgang Mikroprozessortechnik, Seite S 102.)

Vor dem Rücksprung zum Label LOOP, bei dem das neue Bitmuster über den PIO-Port B ausgegeben wird, wartet das Programm einen Augenblick in der WAIT-Routine; die gerade angeschaltete Leuchtdiode wird also erst nach einer kurzen Verzögerung wieder ausgeschaltet.

Erst nach dieser Wartezeit wird das neue Bitmuster beim Label LOOP ausgegeben, und damit wird die in der Leuchtdioden-Zeile links nächstfolgende Leuchtdiode angeschaltet.

In der beschriebenen Art läuft das Programm endlos in der beim Label LOOP beginnenden Schleife, die aus nur vier Befehlen besteht. Bei jedem Schleifen-Durchlauf wird die zum geschobenen 1-Bit gehörende Leuchtdiode eingeschaltet. Alle anderen Leuchtdioden werden mit einem Ø-Bit angesteuert und sind dunkel.

An dieser kurzen Befehlsfolge ist nun tatsächlich nichts Besonderes. Das Prinzip der WAIT-Routine haben wir Ihnen bereits auf den Seiten S 17 und S 18 erläutert, und die vier Befehle zur Programmierung des PIO-Ports B ab dem Label INIT brauchen wir jetzt auch nicht mehr zu erklären.

In der Auflistung des Programms finden Sie zwar eine Routine ALARM, die offensichtlich das Klingelzeichen generiert; es ist aber zunächst überhaupt nicht verständlich, wie das Hauptprogramm, das beim Label Start beginnt und mit dem Befehl JR LOOP endet, auf eine Eingabe über den PIO-Port A reagieren kann. Einen Befehl IN A,(PIODA), der Daten vom PIO-Port A einlesen könnte, werden Sie im gesamten Programm vergeblich suchen.

Ursprünglich Portbit 1501-1559
nach 2000 - 2059

Aufgabe S 20.1

Hier ist eine etwas schwierige Aufgabe. Wenn Sie die Lösung nicht gleich finden, dürfen Sie ohne Gewissensbisse unsere Lösung nachlesen.

Überlegen Sie einmal, wie Sie die Abfrage nach einem Alarm-Signal über den PIO-Port A in das Hauptprogramm einbauen könnten. Wenn ein solches Alarm-Signal gefunden wird, dann soll die ALARM-Routine angesprungen werden. Andernfalls soll das Hauptprogramm normal weiterlaufen.

Wenn Sie eine Lösung finden, dann beachten Sie bitte die folgenden Hinweise:

Das Hauptprogramm muß um einige Befehle erweitert werden. Achten Sie bitte darauf, daß sich der Operand des JR LOOP-Befehls am Ende des Hauptprogramms ändert!

Wenn Sie die hier als Aufgabe gestellte Änderung des Programms vornehmen (was Sie auf jeden Fall versuchsweise tun sollten!), dann verschiebt sich die Anfangsadresse der INIT-Routine. Vergessen Sie

nicht, den Operanden des CALL INIT-Befehls im Hauptprogramm entsprechend zu ändern!

Die INIT-Routine muß im Anschluß an das erweiterte Hauptprogramm neu eingetastet werden. Auf das Setzen des Interrupt-Vektors für den PIO-Port A und alle darauf folgenden Befehle der INIT-Routine kann verzichtet werden. Programmieren Sie den RET-Befehl, der die INIT-Routine abschließt, im Anschluß an die Programmierung des *I/O Register Control Words*. Die WAIT-Routine und die ALARM-Routine können dann bei ihren ursprünglichen Adressen stehen bleiben.

Das Unterprogramm ALARM schließt in unserer Programm-Auflistung mit den beiden Befehlen EI und RETI ab. Was diese beiden Befehle bewirken, werden wir nachher erläutern. – Wenn die ALARM-Routine mit einem CALL-Befehl aufgerufen wird, dann müssen die Befehle EI und RETI durch den Befehl RET mit dem Operationscode C9 ersetzt werden.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 4.

Rufen Sie bitte in dem als Lösung der vorstehenden Aufgabe entworfenen Programm mehrmals nacheinander durch sehr kurze Betätigungen der zum Port A gehörenden Tasten Nr. 0 oder Nr. 1 die ALARM-Routine auf! Dabei werden Sie mit Sicherheit feststellen, daß Ihr Aufruf das eine oder das andere Mal vom Programm einfach ignoriert wird, obwohl Sie am Aufleuchten der zur betätigten Taste gehörenden Leuchtdiode eindeutig erkennen können, daß die Taste die Eingabe richtig vermerkt hat. Warum reagiert das Programm auf diese Eingaben manchmal nicht?

Es ist ganz einfach: Das Programm kann die Betätigung einer Taste nur mit dem Befehl IN A,(PIODA) feststellen. Nehmen Sie an, Sie betätigen eine Taste kurz nach dem Einschalten einer Leuchtdiode, wenn der IN A,(PIODA)-Befehl bereits ausgeführt ist. Das Programm springt dann in die WAIT-Routine. Während des Ablaufs dieser WAIT-Routine lassen Sie die Taste aber bereits wieder los. In diesem Fall wird während der Tasten-Betätigung kein IN A,(PIODA)-Befehl ausgeführt. Das Programm kann dann die Tasten-Betätigung nicht erkennen und übersieht sie einfach: Der Alarm wird nicht ausgelöst. Nur wenn Sie die Taste solange festhalten, bis der nächste IN A,(PIODA) ausgeführt wird, kann die Betätigung der Taste erkannt werden.

Ändern Sie das Programm wieder so, wie Sie es ursprünglich eingegeben haben (Seiten L 23 bis L 26). Wenn Sie jetzt eine der Tasten zu beliebiger Zeit sehr kurzzeitig betätigen, dann wird die Auslösung des Alarms mit Sicherheit niemals übersehen.

Der Grund dafür muß eindeutig sein, daß das Programm jederzeit – und nicht nur während der Ausführung eines bestimmten Befehls – in der Lage ist, die den Alarm auslösende Taste zu erkennen und daraufhin die ALARM-Routine anzuspringen: Der normale Programmablauf kann durch die Betätigung der Taste jederzeit für den Ansprung der ALARM-Routine unterbrochen werden.

Das ist es: Das Programm ist für eine Unterbrechung, für einen **Interrupt** empfänglich gemacht worden.

S
22

Aufruf einer Interrupt-Service-Routine

Sehen Sie sich bitte noch einmal das Bild H19.1 an, in dem wir die Zusammenschaltung der Z 80 CPU dargestellt haben. Sie finden dort die Verbindung zwischen je einem Anschluß $\overline{\text{INT}}$ der CPU und der PIO. Die Bezeichnung INT ist die Abkürzung der englischen Bezeichnung *INTerrupt Request.* Wörtlich bedeutet das: Interrupt-Anforderung. Mit dem Querstrich über der Bezeichnung ist angedeutet, daß das mit dieser Verbindung von der PIO zur CPU übertragene Signal *active LOW* ist. Es hat normalerweise den Wert 1. Wenn die PIO ein Ø-Signal auf diese Leitung schaltet, dann teilt sie damit der CPU mit, daß eine Peripherie-Einheit eine Programm-Unterbrechung wünscht und daß jetzt eine dieser Unterbrechung zugeordnete Interrupt-Service-Routine (vgl. Seite S4) ablaufen soll.

Ob die CPU diesem Wunsch nach einer Programm-Unterbrechung nachkommt, hängt davon ab, ob sie dazu per Programm ausdrücklich ermächtigt worden ist oder nicht. Der Befehlssatz des Z 80 enthält dafür einen speziellen Befehl **E***nable* **I***nterrupts* (Lasse Interrupts zu), der mit dem mnemonischen Code EI abgekürzt wird. Solange in einem Programm dieser Befehl nicht ausgeführt worden ist, kann die PIO so oft einen Interrupt anfordern, wie sie will: Die CPU wird diese Anforderung einfach übersehen.

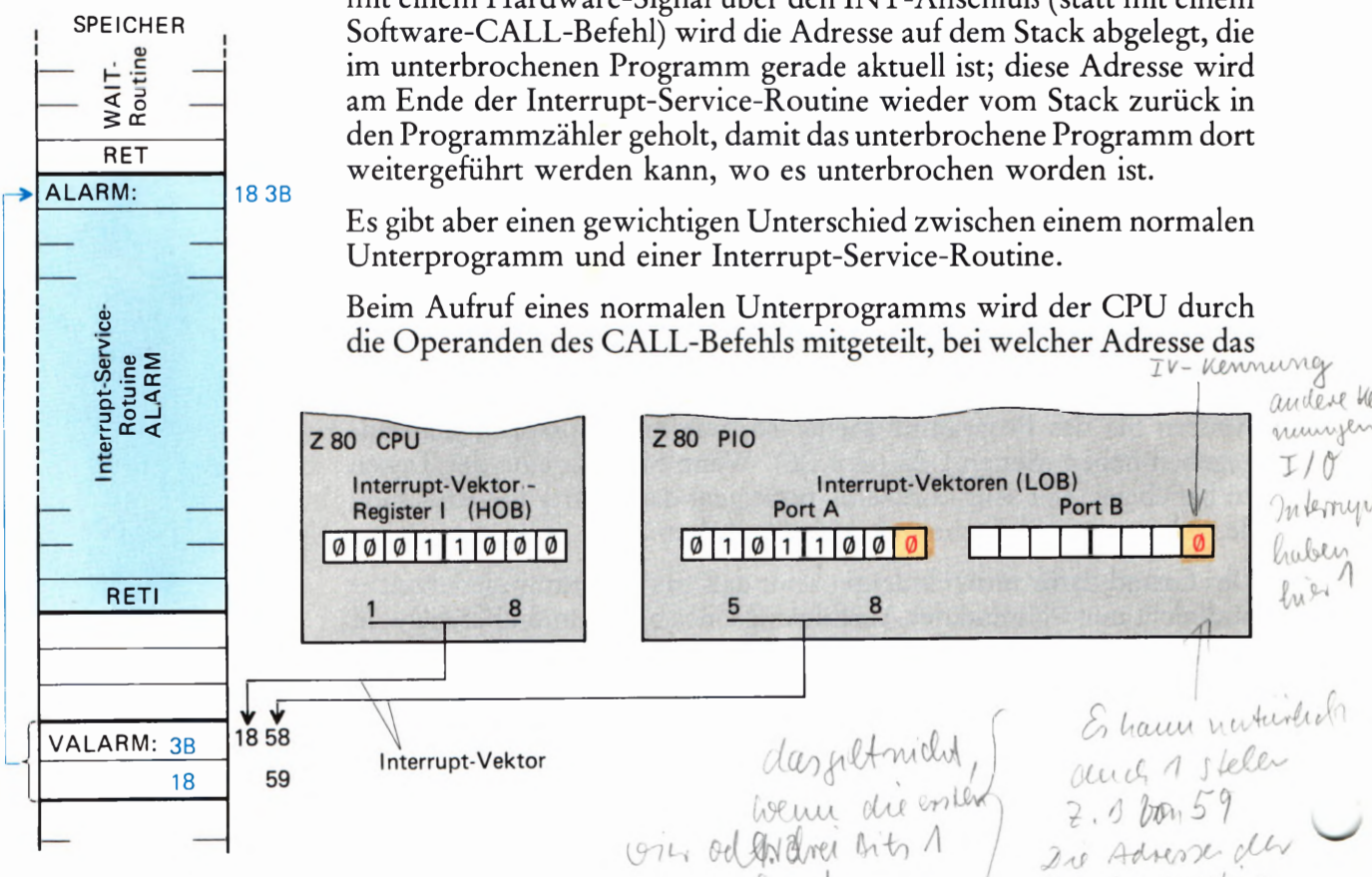
Wir wollen Ihnen an dieser Stelle auch gleich den Befehl vorstellen, mit dem der EI-Befehl widerrufen werden kann. Er heißt **D***isable* **I***nterrupts* (sperre Interrupt-Anforderungen) und wird mit dem mnemonischen Code DI abgekürzt.

Eine Interrupt-Service-Routine hat im Prinzip die gleiche Funktion wie ein normales Unterprogramm. Beim Aufruf einer solchen Routine mit einem Hardware-Signal über den $\overline{\text{INT}}$-Anschluß (statt mit einem Software-CALL-Befehl) wird die Adresse auf dem Stack abgelegt, die im unterbrochenen Programm gerade aktuell ist; diese Adresse wird am Ende der Interrupt-Service-Routine wieder vom Stack zurück in den Programmzähler geholt, damit das unterbrochene Programm dort weitergeführt werden kann, wo es unterbrochen worden ist.

Es gibt aber einen gewichtigen Unterschied zwischen einem normalen Unterprogramm und einer Interrupt-Service-Routine.

Beim Aufruf eines normalen Unterprogramms wird der CPU durch die Operanden des CALL-Befehls mitgeteilt, bei welcher Adresse das

Bild S22.1
Der Interrupt-Vektor zeigt auf die erste der Ablage-Adressen für die Startadresse der Interrupt-Service-Routine. Das HOB steht im I-Register der CPU; das LOB liefert die PIO.



jetzt auszuführende Unterprogramm steht. Das ist beim Aufruf einer Interrupt-Service-Routine natürlich nicht möglich, denn es ist ja nicht vorauszusehen, welcher Befehl vom Programm gerade bearbeitet wird, wenn die Peripherie einen Interrupt anfordert.

Beim Z 80 Mikroprozessor wird zur Lösung dieses Problems eine einfache Methode angewendet: Die beiden Bytes der Startadresse der Interrupt-Service-Routine werden in zwei vereinbarten Speicherzellen abgelegt: Das LOB der Startadresse in der Speicherzelle mit der niedrigeren Adresse und das HOB in der Speicherzelle mit der höheren Adresse. Dort holt sich die CPU dann die Startadresse der abzuarbeitenden Interrupt-Service-Routine ab, wenn die Peripherie eine Programmunterbrechung wünscht.

Wir wollen Sie bereits jetzt auf eine wichtige Eigenschaft des Interrupt-Systems des Z 80 hinweisen:

> Das LOB der Startadresse einer Interrupt-Service-Routine muß beim Z 80-Mikroprozessor immer bei einer geraden (also ohne Rest durch zwei teilbaren) Adresse abgelegt werden. Im Bitmuster einer solchen Adresse hat das Bit Nr. 0 den Wert Ø.

Sehr schön, aber wie wird der CPU mitgeteilt, wo sie die Adresse der Interrupt-Service-Routine finden kann?

Wir haben vorhin gesagt, daß einer von der Peripherie gewünschten Programm-Unterbrechung eine bestimmte Interrupt-Service-Routine zugeordnet ist. Das bedeutet: Zusammen mit der Interrupt-Anforderung muß der CPU mitgeteilt werden, wo sie im Speicher die Startadresse der Interrupt-Service-Routine finden kann.

Sehen Sie sich bitte das Bild S 22.1 an! Die Darstellung bezieht sich auf das Lauflicht-Programm, das wir ab der Seite L 23 aufgelistet haben. Die Interrupt-Service-Routine ALARM beginnt bei der Speicher-Adresse 183B. Das LOB dieser Adresse haben wir im Speicher bei der Adresse 1858H abgelegt; das HOB der Startadresse steht in der Speicherzelle mit der Adresse 1859H.

Rechts im Bild haben wir die hier interessierenden Register der Z 80 CPU und der Z 80 PIO dargestellt. – Die CPU enthält ein Register mit der Bezeichnung I, um das Sie sich wahrscheinlich bisher noch nie gekümmert haben. (Vgl. Lehrgang Mikroprozessortechnik, Seite T 2). Dieses acht Bit „breite" Register dient zur Aufnahme des HOB eines **Interrupt-Vektors,** der auf die erste der beiden Speicherzellen zeigt, in denen die Startadresse einer Interrupt-Service-Routine abgelegt ist (1858H).

Das Bild S 22.1 zeigt deutlich, wo das LOB der ersten der beiden Ablage-Adressen (1858H) herkommt: Es wird von der Z 80 PIO geliefert. Wir haben Sie eben darauf hingewiesen, daß dieses Byte immer eine gerade Zahl darstellen muß: Das Bit Nr. 0 dieses Bytes muß den Wert Ø haben. In unserer Darstellung haben wir aus nachher ersichtlichem Grund dieses Ø-Bit durch rote Darstellung hervorgehoben.

Der Inhalt des I-Registers der CPU und ein vom Peripherie-Baustein zusammen mit einer Interrupt-Anforderung geliefertes Byte werden

als **Interrupt-Vektor** bezeichnet. Dieser Vektor zeigt auf die niedrigere der beiden Ablage-Adressen, in denen die Startadresse einer Interrupt-Service-Routine abgelegt ist.

Das Bild S22.1 erläutert bereits in eindeutiger Form den gesamten Interrupt-Mechanismus des Z 80-Systems. Sie sollten sich das so merken:

S

24

Die Startadresse einer Interrupt-Service-Routine wird beim Z 80 Mikroprozessor in zwei aufeinanderfolgenden Speicherzellen abgelegt. Im Bitmuster der niedrigeren Adresse der Ablage-Speicherzellen hat das Bit Nr. 0 stets den Wert Ø. Bei dieser Adresse ist das LOB der Startadresse der Interrupt-Service-Routine abgelegt.

Das HOB der Ablage-Adresse steht im I-Register der CPU; das LOB der Ablage-Adresse wird zusammen mit der Interrupt-Anforderung vom Peripherie-Baustein geliefert.

Wir brauchen uns um den Hardware-Mechanismus beim Aufruf einer Interrupt-Service-Routine weiter keine Gedanken zu machen. Es genügt die Feststellung, daß bei einer Interrupt-Anforderung der Inhalt des I-Registers in der CPU zusammen mit einem Interrupt-Vektor in der PIO für die CPU die Information bilden, wo sie die Startadresse der Interrupt-Service-Routine abholen kann.

Bitte beachten Sie, daß dieser Mechanismus nur dann in Kraft tritt, wenn die CPU mit einem EI-Befehl vorher ausdrücklich ermächtigt worden ist, auf Interrupt-Anforderungen zu reagieren. Andernfalls wird eine Interrupt-Anforderung einfach übersehen.

Aufgabe S24.1

a) Im Lauflicht-Programm haben wir die Interrupt-Service-Routine ALARM gleich im Anschluß an die WAIT-Routine ab der Adresse 183B programmiert. Welche Änderung ergäbe sich in der Darstellung im Bild S22.1, wenn die ALARM-Routine bei der Adresse 18F7 beginnt?

b) Außer der Verlegung des Beginns der ALARM-Routine auf die Adresse 18F7 sollen aus irgendwelchen Gründen die Ablage-Speicherzellen für die Startadresse der Interrupt-Service-Routine ALARM bei den Adressen 1926H und 1927H liegen.

Welche zusätzlichen Änderungen ergeben sich dann in der Darstellung in Bild S22.1?

c) Dürfte die Startadresse der ALARM-Routine auch in den Speicherzellen mit den Adressen 1925H (LOB) und 1926H (HOB) abgelegt werden, wenn die CPU auf diese Ablage zum Aufruf der Interrupt-Service-Routine zugreifen soll?

Begründen Sie bitte Ihre Antwort!

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü4.

Die Interrupt-Programmierung der Z 80 PIO

Wenn ein Z 80-System so eingerichtet werden soll, daß es auf Interrupt-Anforderungen reagiert, dann müssen drei Maßnahmen getroffen werden:

1. Die Hardware muß so eingerichtet sein, daß der CPU ein Interrupt-Signal an den $\overline{\text{INT}}$-Anschluß übermittelt wird. Dieses Signal kann im einfachsten Fall von der Peripherie selbst kommen. Üblicherweise wird es aber von einem Interface-Baustein generiert. Der Interface-Baustein wiederum kann so programmiert werden, daß beliebige Peripherie-Signale oder Signal-Kombinationen einen Interrupt auslösen.

2. Die CPU muß so programmiert werden, daß sie Interrupt-Anforderungen ausführen kann.

   Voraussetzung für das Auffinden einer Interrupt-Service-Routine ist es, daß im I-Register der CPU das HOB der Startadresse dieser Routine abgelegt ist.

   Die CPU kann auf Interrupt-Anforderungen auf verschiedene Weise reagieren. Sie muß mit einem speziellen Befehl programmiert werden, damit sie eine Programm-Unterbrechung so ausführt, wie es für den jeweils vorliegenden Fall zweckmäßig ist.

   Schließlich muß mit dem bereits vorgestellten EI-Befehl dafür gesorgt werden, daß die CPU Interrupt-Anforderungen wahrnehmen kann.

3. Der Peripherie-Baustein muß so programmiert werden, daß er unter definierten Bedingungen ein Interrupt-Signal an den $\overline{\text{INT}}$-Anschluß der CPU liefert.

Die im Punkt 2 angedeutete Interrupt-Programmierung der CPU werden wir Ihnen im nächsten Abschnitt noch kurz vorstellen. Hier wollen wir nur anmerken, daß im Lauflicht-Programm (Seite L 22) die letzten vier Anweisungen der INIT-Routine die Interrupt-Programmierung der CPU besorgen.

Der dritte Punkt unserer Aufstellung ist für uns in diesem Zusammenhang natürlich besonders interessant. Wir wollen uns ansehen, welche Möglichkeiten die Z 80 PIO zur Generierung eines Interrupt-Anforderungssignals zur Verfügung stellt und wie von diesen Möglichkeiten Gebrauch gemacht wird.

Im Lauflicht-Programm haben wir beide Ports der PIO zum Arbeiten in der Betriebsart 3 (Mode 3) programmiert (vgl. die INIT-Routine auf der Seite L 22). – Sehen Sie sich bitte noch einmal das Programmierungsschema für die PIO auf der Seite S 8 an! Grundsätzlich muß für jeden Port zuerst mit dem *Mode Control Word* die Betriebsart festgelegt werden. In der Betriebsart 3 folgt dann bei beiden Ports das *I/O Register Control Word*, das die einzelnen Port-Anschlüsse zu Eingängen oder Ausgängen machen kann. Diese beiden Befehle haben wir bereits erläutert. Sie können sie in der INIT-Routine des Lauflicht-Programms leicht identifizieren.

Die auf der Seite S 8 unter der Betriebsart 3 nach dem *I/O Register Control Word* folgenden vier Anweisungen dienen sämtlich der Interrupt-

Programmierung der PIO. Wir haben uns um diese Anweisungen bisher nicht gekümmert, weil wir ja noch nicht mit Interrupts gearbeitet haben. In der INIT-Routine des Lauflicht-Programms finden Sie jedoch drei der vier auf der Seite S 8 aufgelisteten Interrupt-Anweisungen bei der Programmierung des PIO-Ports A. Die letzte der auf der Seite S 8 aufgelisteten Anweisungen *(Interrupt Disable Word)* wird nur selten verwendet. Sie kann die Ausgabe eines Interrupt-Anforderungs-Signals an die CPU auch dann verhindern, wenn die Peripherie entsprechende Signale aussendet.

Wir wollen uns die Interrupt-Programmier-Anweisungen für die Z 80 PIO der Reihe nach genau ansehen. Die Wirkungen dieser Anweisungen können Sie dann am Lauflicht-Programm teilweise sofort ausprobieren. Zuvor müssen wir aber die grundsätzliche Programmierung der PIO für diesen Versuch noch einmal betrachten.

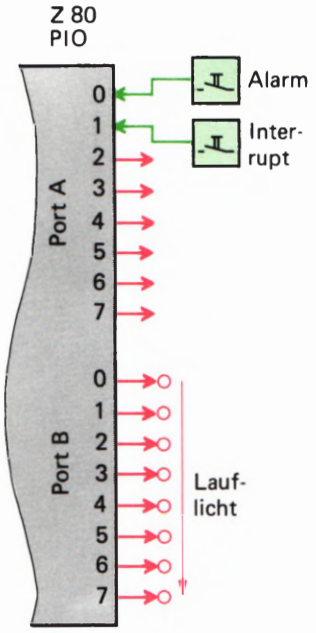


Bild S 26.1
Im Lauflicht-Programm werden die Leuchtdioden über den Port B angesteuert. Über den Port A werden die Interrupt-Anforderungen eingegeben.

Bei der Ausführung des Versuchs haben Sie bereits festgestellt, welche Funktionen den beiden PIO-Ports zugeordnet sind. Im Bild S 26.1 sind diese Funktionen zusammengestellt.

Über den Port B wird das Lauflicht gesteuert. Da alle acht Leuchtdioden am Lauflicht teilnehmen, werden mit dem *I/O Register Control Word* für diesen Port auch alle acht Anschlüsse als Ausgänge programmiert. Mit Interrupt hat dieser Port nichts zu schaffen. Die auf der Seite S 8 aufgelisteten Interrupt-Anweisungen brauchen also bei der Programmierung des Ports B nicht berücksichtigt zu werden.

Anders ist es mit dem Port A. Dieser Port hat mit dem eigentlichen Lauflicht-Programm nicht das geringste zu tun. Die Steuerung des Lauflichts, die vom Hauptprogramm besorgt wird, tut so, als gäbe es den Port A gar nicht.

Der Port A wird nur für die Eingabe der Alarm-Signale verwendet. Mit dem für diesen Port programmierten *I/O Register Control Word* haben wir die beiden Anschlüsse Nr. 0 und Nr. 1 zu Eingängen gemacht und dementsprechend auch die Brücken auf die Stiftleisten zum Port A gesteckt (vgl. das Bild S 19.1). Im Zusammenhang mit dem nachher vorgestellten *Mask Control Word* ergibt sich damit die Möglichkeit, wahlweise Alarm-Interrupts sowohl über die Taste Nr. 0 als auch über die Taste Nr. 1 des Ports A auszulösen. Überzeugen Sie sich davon im Versuch!

Die Anschlüsse Nr. 2 bis Nr. 7 des Ports A sind als Ausgänge programmiert. Im Versuch haben wir die zugehörigen Tasten nicht durch steckbare Brücken auf den Stiftleisten aktiviert (vgl. das Bild H 12.1).

Machen Sie bitte den folgenden Versuch:

Versuch S 26.1

Zuordnung der Interrupt-Eingänge

Ändern Sie in dem auf den Seiten L 23 bis L 26 aufgelisteten Lauflicht-Programm das Byte bei der Adresse 181A vom Wert Ø3H in den Wert FF! Damit werden sämtliche acht Anschlüsse des Ports A als Eingänge programmiert. – Bestücken Sie alle acht Stiftpaare der Stiftleiste zum Port A mit steckbaren Brücken. Jetzt sind sämtliche acht Tasten zum Port A aktiviert. Wenn Sie eine der Tasten betätigen, leuchtet die zugehörige Leuchtdiode.

Starten Sie das Programm und stellen Sie fest, mit welchen der zum Port A gehörenden Tasten Sie einen Alarm-Interrupt auslösen können! Der Versuch zeigt, daß nach wie vor nur mit den beiden Tasten Nr. 0 und Nr. 1 ein Interrupt verursacht werden kann.

Sie werden nachher sehen, daß für diese Zuordnung der Interrupt-Eingänge das Interrupt-*Mask Control Word* zuständig ist.

Jetzt können wir darangehen, uns die Anweisungen anzusehen, mit denen die Eingabe von Alarm-Interrupts über den Port A ermöglicht wird.

Bei der CPU ist die erste Software-Maßnahme zur Interrupt-Programmierung die Eintragung des Interrupt-Vektor-HOBs in das I-Register (Seite S 25). Ganz entsprechend muß bei der PIO vorrangig das Interrupt-Vektor-LOB eingetragen werden. Dieses Byte wird als *Interrupt Vector Word* bezeichnet. Es muß der PIO in unserem Programm von der CPU an die Befehlsadresse 82H des Ports A (PIOBA, vgl. die Seiten H 21 und S 13) übermittelt werden.

Die Bezeichnung *Interrupt Vector Word* ist eigentlich irreführend. Der Interrupt-Vektor zeigt auf die untere der beiden Ablage-Speicherzellen, in denen die Startadresse der Interrupt-Service-Routine abgelegt ist (vgl. das Bild S 22.1 und die Seite S 24). Der Interrupt-Vektor ist also in Wirklichkeit eine Sechzehn-Bit-(Zwei-Byte-)Adresse. Die CPU schickt als *Interrupt Vector Word* nur das LOB dieser Adresse an die PIO; das HOB des Interrupts-Vektors wird im I-Register der CPU abgelegt.

Die Befehlsfolge zur Eintragung des *Interrupt Vector Words* für den PIO-Port A sieht dann (am Beispiel unseres Programms) so aus:

```
IVLOB   EQU   LOW VALARM    ;Interrupt-Vektor LOB
PIOBA   EQU   82H           ;PIO-Befehls-Adresse für
                             den Port A

        LD    A,IVLOB       ;LOB des Interrupt-Vektors
                             in den Akku
        OUT   (PIOBA),A     ;  und an die PIO-Befehls-
                             adresse
```

(Lesen Sie bitte über die EQU-Anweisung noch einmal auf der Seite L 2 nach!)

Wir haben vorhin gesagt, daß die Übermittlung des Interrupts-Vektor-LOBs an die PIO vorrangig ist. Gemeint ist damit nicht, daß das *Interrupt Vector Word* unbedingt als erste der Interrupt-Anweisungen an die PIO geschickt werden muß. Mit vorrangig haben wir die Wichtigkeit dieser Information unterstrichen. Es ist durchaus möglich, dieses Byte als letztes bei der Initialisierung des betreffenden PIO-Ports zu übermitteln. Nur darf es eben auf keinen Fall vergessen werden.

Wenn Sie die INIT-Routine im Lauflicht-Programm ansehen, dann fällt Ihnen sicher auf, daß die auf der Seite S 8 zusammengestellten Programmier-Anweisungen der PIO sämtlich in genau der gleichen Form nacheinander übermittelt werden: Das betreffende Byte wird in den Akkumulator der CPU geladen und von dort mit einem OUT-Befehl an die Adresse 82H des Ports PIOBA geschickt.

S

28

Offenbar ist die Reihenfolge, in der die Anweisungen an die PIO geschickt werden, nicht immer wichtig. Die Frage ist, wie die PIO dann merken kann, welche Bedeutung die einzelnen Bytes haben, die ihr da geschickt werden.

Sehen Sie sich bitte noch einmal den Aufbau des *Mode Control Words* in Bild S 6.1 an! Die niederwertigen vier Bits haben in diesem Byte immer die Werte 1. Durch diese Werte 1 ist das *Mode Control Word* eindeutig als solches gekennzeichnet; es hat also in der sedezimalen Darstellung stets die Form **XF.**

Die Identifizierung des *I/O Register Control Words* macht der PIO keine Schwierigkeiten. Sie haben auf der Seite S 8 gelesen, daß dieses Steuerwort immer mit Sicherheit dem *Mode Control Word* folgt.

Wie sieht es nun mit der Erkennbarkeit des *Interrupt Vector Words* aus? Sehen Sie sich das Bild S 22.1 an! Wir haben dort innerhalb der Z 80 PIO das Bit Nr. 0 des Interrupt-Vektor-LOBs rot mit dem Wert Ø eingetragen. Dieser Wert Ø des Bits Nr. 0 ist es, der ein *Interrupt Vector Word* als solches eindeutig kennzeichnet.

Jetzt haben Sie auch die Erklärung für die zunächst ziemlich willkürlich erscheinende Forderung, daß die Adresse der unteren beiden Speicherzellen, in denen die Startadresse der Interrupt-Service-Routine abgelegt wird, immer ohne Rest durch zwei teilbar sein muß (vgl. Seite S 23). Diese Forderung wird dann erfüllt, wenn das Bit Nr. 0 des Interrupt-Vektors den Wert Ø hat. Und das wiederum ist zur Kennzeichnung des der PIO übermittelten *Interrupt Vector Words* notwendig.

Sowohl in unserer Aufstellung auf der Seite S 8 als auch in der INIT-Routine des Lauflicht-Programms folgt auf die Programmierung des Interrupt-Vektor-LOBs das *Interrupt Control Word*. Mit diesem Byte werden die Bedingungen festgelegt, unter denen die Peripherie einen Interrupt auslösen kann. Es ist sicher das interessanteste der PIO-Interrupt-Anweisungen. Sie werden sich in einigen Versuchen ansehen können, welche Möglichkeiten die Variationen dieses Bytes bieten, vor allem im Zusammenhang mit einem nachfolgenden Byte für die Interrupt-Maske.

Das Bild S 28.1 zeigt den Bit-Aufbau des Steuerworts für den Interrupt. Ähnlich wie beim *Mode Control Word* können wir auch hier die niederwertigen vier Bits vorweg abtun: Diese Bits mit dem Muster Ø111 bilden die Kennung des *Interrupt Control Words*. In der sedezimalen Darstellung hat es immer die Form X7H.

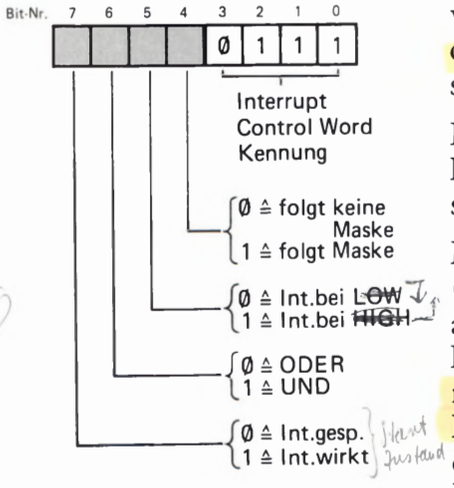


Bild S 28.1
Bitmuster des Interrupt-Steuerworts. Die Bits Nr. 0 bis Nr. 3 bilden die Kennung.

Maßgeblich für die Interrupt-Bedingungen sind die vier höherwertigen Bits Nr. 4 bis Nr. 7, deren Bedeutungen in das Bild S 28.1 eingetragen sind.

Mit dem Wert des Bits Nr. 4 wird festgelegt, ob die PIO das *Interrupt Control Word* als Ein- oder als Zwei-Byte-Befehl auffaßt, ob die PIO also nach diesem Byte ein weiteres, nicht besonders gekennzeichnetes Byte erwarten soll oder nicht. Wenn das Bit Nr. 4 den Wert 1 hat, dann muß gleich anschließend ein weiteres Byte abgeschickt werden, das die Bedeutung einer Interrupt-Maske hat. Sie werden nachher sehen, was eine solche Maske bewirkt. – Beim Wert 0 des Bits Nr. 4 erwartet die PIO keine anschließende Interrupt-Maske.

Die Bedeutung des Bits Nr. 5 wollen wir zunächst übergehen. Sie können sich in einem Versuch ansehen, was das Bit Nr. 6 bewirkt.

Versuch S28.1

Die Verknüpfung von Interrupt-Signalen

Ersetzen Sie im Lauflicht-Programm das *Interrupt Control Word* B7 (Bitmuster 1Ø11 Ø111) bei der Adresse 1822H durch das Byte F7 mit dem Bitmuster 1111 Ø111! Starten Sie das Programm und versuchen Sie, durch die Betätigung einer der zum Port A gehörenden Tasten Nr. 0 oder Nr. 1 einen Alarm-Interrupt auszulösen! Der Versuch wird nicht gelingen.

Betätigen Sie die beiden Tasten Nr. 0 und Nr. 1 gleichzeitig! Jetzt ertönt das Alarm-Signal.

In der bisherigen Programm-Version konnte ein Alarm über die Betätigung der Taste Nr. 0 **ODER** über die Betätigung der Taste Nr. 1 ausgelöst werden. Die ODER-Verknüpfung der beiden Interrupt-Eingänge haben wir durch die Umprogrammierung des Bits Nr. 6 im *Interrupt Control Word* offensichtlich in eine UND-Verknüpfung umgewandelt: Ein Alarm wird jetzt nur dann ausgelöst, wenn an den einen **UND** (gleichzeitig) an den anderen Interrupt-Eingang ein 1-Signal geschaltet wird.

Dieser Versuch zeigt die Bedeutung des Bits Nr. 6 im *Interrupt Control Word* ohne weitere Erläuterungen. Es soll jedoch noch angemerkt werden, daß diese UND-Verknüpfung auch dann gilt, wenn die PIO für mehr als zwei Interrupt-Eingänge programmiert wird. Wenn also z. B. vier Interrupt-Eingänge programmiert werden, dann wird ein Interrupt bei programmierter UND-Verknüpfung nur dann ausgelöst, wenn an alle vier Eingänge gleichzeitig Interrupt-Anforderungen gemeldet werden.

Jetzt können Sie sich die Bedeutung des Bits Nr. 5 im *Interrupt Control Word* ansehen. Am einfachsten ist es wieder, einen entsprechenden Versuch zu machen.

Versuch S29.1

Interrupt mit Ø-Signalen

Ändern Sie im Lauflicht-Programm das *Interrupt Control Word* bei der Adresse 1822H vom ursprünglichen Wert B7 (Bitmuster 1Ø11 Ø111 mit ODER-verknüpften Interrupt-Signalen) auf den Wert 97H mit dem Bitmuster 1ØØ1 Ø111. Das Bit Nr. 5, um das es hier geht, hat jetzt also den Wert Ø.

Starten Sie das Programm und versuchen Sie, durch Betätigungen der Tasten Nr. 0 und Nr. 1 einen Interrupt auszulösen! Wiederholen Sie den Versuch mehrfach!

Haben Sie herausgefunden, unter welchen Bedingungen ein Interrupt ausgelöst wird? Versuchen Sie, diese Bedingungen in einem Satz zu formulieren!

Das Ergebnis des Versuchs ist zunächst verblüffend. Nach dem Start des Programms leuchten die beiden hier interessierenden Leuchtdioden nicht, weil ja zunächst keine Taste betätigt ist. Die Betätigung einer einzelnen Taste löst keinen Interrupt aus, obwohl mit dem Wert Ø des Bits Nr. 6 eine ODER-Verknüpfung der Interrupt-Signale programmiert

ist. Auch die gleichzeitige Betätigung beider Tasten bringt keinen Erfolg. Sobald Sie aber eine der beiden gleichzeitig betätigten Tasten loslassen, ertönt das Alarm-Signal. Was bedeutet das?

Wenn Sie davon ausgehen, daß bei der jetzt verwendeten Programmierung die Werte 1 der Interrupt-Signale den Normalfall darstellen, dann wird die Sache durchsichtig: Ein Interrupt wird dann ausgelöst, wenn das eine ODER das andere Interrupt-Signal den Wert Ø annimmt.

Diese Erkenntnis macht die Wirkung des Bits Nr. 5 im *Interrupt Control Word* deutlich: Beim Wert 1 dieses Bits werden Interrupts beim Ø-1-Übergang eines Interrupt-Signals ausgelöst; beim Wert Ø dieses Bits werden Interrupts beim 1-Ø-Übergang eines Interrupt-Signals ausgelöst.

Es wird auch klar, warum nach dem Start des Lauflichts bei 1-Ø-programmierten Interrupt-Signalen nicht gleich ein Interrupt erfolgt, obwohl doch zunächst keine der Interrupt-Tasten betätigt ist, also beide Interrupt-Signale die Werte Ø haben: Der Interrupt wird durch den Signalwechsel vom Wert 1 auf den Wert Ø ausgelöst, und dieser Wechsel hat noch nicht stattgefunden.

Wir wollen es Ihrem (hoffentlich vorhandenen) Spieltrieb überlassen, herauszufinden, welche Möglichkeiten sich für verschiedene Interrupt-Strukturen, z.B. durch die Kombination der Werte Ø des Bits Nr. 5 und 1 des Bits Nr. 6, ergeben. Beim Spielen mit diesen Bit-Werten ergeben sich für technische Anwendungen ungeahnte Möglichkeiten. Besonders interessant wird das in Verbindung mit dem gleich noch vorzustellenden *Mask Control Word.*

Zunächst können Sie sich aber noch die Wirkung des Bits Nr. 7 im *Interrupt Control Word* in einem Versuch ansehen.

Versuch S 30.1

Sperren der PIO für Interrupts

Programmieren Sie bitte im Lauflicht-Programm bei der Adresse 1822H den Wert 37H (Bitmuster ØØ11 Ø111) für das *Interrupt Control Word.* Mit dem Wert 1 des Bits Nr. 5 werden Ø-1-Übergänge der Interrupt-Signale wirksam gemacht; der Wert Ø des Bits Nr. 6 bewirkt die ODER-Verknüpfung der Interrupt-Signale. Bis auf das Bit Nr. 7 mit dem Wert Ø haben alle Bits die ursprünglichen programmierten Werte.

Starten Sie das Programm und versuchen Sie, einen Interrupt auszulösen. Es wird Ihnen nicht gelingen.

Ändern Sie den Wert des Bits Nr. 7 bei der Adresse 1822H wieder auf 1 (Byte B7, Bitmuster 1Ø11 Ø111)! Nach dem Start des Programms stellen Sie fest, daß die Interrupts wieder funktionieren.

Sie sehen, daß mit dem Wert Ø des Bits Nr. 7 im *Interrupt Control Word* die PIO an der Weitergabe eines INT-Signals (Bild H 19.1, Seite S 22) gehindert wird. Der Wert 1 dieses Bits macht die Weitergabe und damit die Auslösung eines Interrupts möglich.

In der INIT-Routine des Lauflicht-Programms hat das Bit Nr. 4 im *Interrupt Control Word* den Wert 1. Entsprechend der Bedeutung dieses Bits (vgl. das Bild S 28.1) erwartet die PIO unmittelbar anschlie-

ßend ein Byte als *Mask Control Word*. Mit diesem Steuerwort können von der Peripherie an die PIO gelieferte Signale in die Lage versetzt werden, Interrupt-Anforderungen an die CPU zu schicken.

Bedingung dafür, daß ein PIO-Port-Anschluß zum Interrupt-Anschluß wird, ist natürlich, daß dieser Anschluß vorher mit dem *I/O Register Control Word* als Eingang definiert worden ist.

Die Arbeitsweise des *Mask Control Words* ist ganz ähnlich der des *I/O Register Control Words:* Jedes Bit ist entsprechend seiner Numerierung dem mit der gleichen Nummer bezeichneten Port-Anschluß der PIO zugeordnet. Ein Ø-Bit im *Mask Control Word* definiert den zugehörigen Port-Anschluß als Interrupt-Eingang; ein 1-Bit läßt den betreffenden Anschluß unbeeinflußt.

In der INIT-Routine des Lauflicht-Programms haben wir das *Mask Control Word* einfach als Interrupt Maske bezeichnet. Es hat das Bitmuster 1111 11ØØ. Mit den beiden Ø-Bits Nr. 0 und Nr. 1 werden die zu den Tasten Nr. 0 und Nr. 1 gehörenden Anschlüsse des Ports A als Interrupt-Eingänge definiert.

Sie haben nun die wichtigsten Steuerworte zur Programmierung der Z 80 PIO in der Betriebsart 3 einschließlich der Steuerworte zur Interrupt-Programmierung kennengelernt. Es steht noch die kurze Erläuterung des *Interrupt-Disable Words* aus. Trotzdem können Sie sicher bereits jetzt die folgende Aufgabe lösen:

Aufgabe S 31.1

Im Lauflicht-Programm entsprechend der Programmliste ab der Seite L 23 soll ein Interrupt nur durch die gleichzeitige Betätigung der zum Port A gehörenden Tasten Nr. 0, Nr. 2 und Nr. 7 ausgelöst werden können. Die übrigen Anschlüsse des Ports A sollen als Ausgänge programmiert werden.

In Bild S 31.1 ist die geforderte Beschaltung des PIO-Ports A zu erkennen.

Schreiben Sie bitte sämtliche Befehle an, die zur hier geforderten Programmierung des Ports A notwendig sind.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 5.

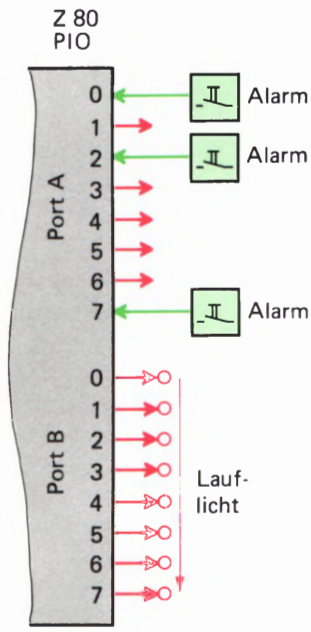


Bild S 31.1
Zu Aufgabe S 31.1: Die Programmierung der PIO für das Lauflicht-Programm soll so geändert werden, daß das hier gezeigte Eingabe-Schema berücksichtigt wird.

Als letzten der Programmierbefehle für die Betriebsart 3 der Z 80 PIO haben wir auf der Seite S 8 das *Interrupt Disable Word* aufgeführt. Es wurde bereits auf der Seite S 26 darauf hingewiesen, daß dieser Befehl verhältnismäßig selten verwendet wird.

Das Bild S 31.2 zeigt die Bedeutung der einzelnen Bits in diesem Befehls-Byte. Die Bits Nr. 0 bis Nr. 3 bilden mit dem Muster ØØ11 die Kennung des *Interrupt Disable Words*. Es hat in der sedezimalen Darstellung immer die Form X3H.

Das englische Wort *disable* (sprich: disäibel) bedeutet: Unbrauchbar bzw. unwirksam machen. Die Übermittlung des Befehlsbytes 83H (Bitmuster 1XXX ØØ11) hat die gleiche Wirkung wie ein *Interrupt Control Word*, in dem das Bit Nr. 7 den Wert 1 hat (vgl. Bild S 28.1).

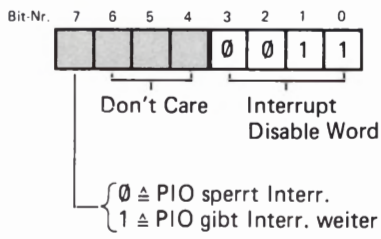


Bild S 31.2
Bitmuster des Interrupt-Disable-Worts. Die Bits Nr. 0 bis Nr. 3 bilden die Kennung.

Durch die Aussendung eines *Interrupt Disable Words* mit dem Wert 1 des Bits Nr. 7 kann die vorangegangene Sperrung einer Interrupt-Anforderung an die CPU wieder aufgehoben werden.

Sie haben jetzt insgesamt drei Methoden kennengelernt, Interrupt-Anforderungen der Peripherie unwirksam und danach auch wieder wirksam zu machen. Zum einen sind das die beiden Möglichkeiten über die PIO-Interrupt-Befehle (Bit Nr. 8 im *Interrupt Control* und im *Interrupt Disable Word*. Die andere Möglichkeit ist die über die Befehle DI mit dem Operationscode F3 und EI mit dem Operationscode FB (vgl. die Seite S 22).

Beachten Sie aber bitte die Unterschiede: Die ersten beiden Möglichkeiten sperren oder öffnen die PIO für die Weitergabe von Interrupt-Wünschen der Peripherie. Davon ist es ganz unabhängig, ob die CPU für die Annahme oder Ablehnung von Interrupt-Anforderungen programmiert ist.

Die dritte Möglichkeit betrifft die CPU selbst. Nach dem Einschalten verhält sie sich zunächst ablehnend gegen Interrupt-Anforderungen und muß mit dem Befehl EI erst einmal aufnahmebereit dafür gemacht werden. Wir werden Ihnen noch zeigen, daß sie nach einem durchgeführten Interrupt sogleich wieder die Abwehrstellung einnimmt und dann jedesmal neu für die Annahme weiterer Interrupt-Anforderungen programmiert werden muß.

Zum Abschluß dieses Kapitels wollen wir noch einmal auf eine wichtige Tatsache hinweisen, die nicht übersehen werden darf. In der Aufstellung auf der Seite S 8 erkennen Sie, daß die hier beschriebene Interrupt-Programmierung der Z 80 PIO nur für die Betriebsart 3 (Mode 3) gilt.

Selbstverständlich kann die Z 80 PIO auch in den Betriebsarten Ø, 1 und 2 Interrupt-Anforderungen über die INT-Leitung an die CPU senden. Die Peripherie meldet dann ihre Interrupt-Wünsche nicht über einen der Daten-Anschlüsse der PIO-Ports an, sondern über die eigens dafür zuständigen STROBE-Anschlüsse (vgl. die Seite S 4 und die Bilder S 3.1 und S 5.1). Wir werden Ihnen für den Interrupt-Betrieb außerhalb der Betriebsart 3 noch ein Beispiel vorstellen.

In der Betriebsart 3 sind die Handshake-Anschlüsse der PIO, zu denen der STROBE-Anschluß gehört, außer Betrieb. Sie haben jedoch sicher bereits festgestellt, daß dafür die Betriebsart 3 außerordentlich vielseitig und leistungsfähig ist.

Ein Morse-Programm mit Interrupts

Wenn Sie den Lehrgang Mikroprozessortechnik durchgearbeitet haben, dann ist Ihnen das in diesem Abschnitt vorgestellte Programm bereits geläufig (Lehrgang Mikroprozessortechnik ab Seite S 93). Dieses Programm ist in seiner ursprünglichen Version im Micro-Professor-System auch ohne angeschlossene Peripherie-Leiterplatte lauffähig.

Uns geht es in diesem Zusammenhang nicht so sehr darum, Ihnen ein neues Programm vorzuführen, obwohl das Programm vielleicht ganz interessant ist. Wir wollen Ihnen an diesem Programm vielmehr Interrupt-Anwendungen vorführen, die wir in den vorhergehenden Abschnitten noch nicht zeigen konnten.

Das Prinzip des Morse-Programms

Das Morse-Programm hat die Aufgabe, im Speicher des Systems abgelegte Buchstaben akustisch in Form von langen und kurzen Zeichen auszugeben. Das Prinzip des Morse-Codes ist es, alphanumerische Zeichen, also Buchstaben und Zahlen sowie Satz- und Sonderzeichen als Kombinationen von bis zu sechs langen und kurzen Zeichen darzustellen.

Die Zuordnung von Buchstaben, Zahlen und Satzzeichen zu den Kombinationen langer und kurzer Zeichen des Morse-Codes haben wir auf der Seite L 30 dargestellt. – Wenn Sie dort die jeweils zur Code-Zuordnung gehörenden Bitmuster und Bytes mit der Zuordnung des ASCII-Codes auf der Seite T 1 vergleichen, dann erkennen Sie, daß wir die als Morsezeichen auszugebenden Buchstaben im Speicher nicht in Form von ASCII-Zeichen ablegen. Das ein alphanumerisches Zeichen darstellende Bitmuster ist der Kombination von langen und kurzen Zeichen des Morse-Codes angepaßt: Ein Ø-Bit ist einem kurzen Zeichen zugeordnet und ein 1-Bit einem langen Zeichen. Dabei werden die Bitmuster – ebenso wie die Morsezeichen – links beginnend betrachtet. Das in unserer Zusammenstellung rot eingetragene 1-Bit stellt keinen Teil eines Morsezeichens mehr dar; es ist ein sogenannter Terminator, der das Ende des Zeichens darstellt.

Das Morse-Programm ist so aufgebaut, daß es zwischen der Ausgabe von zwei Morsezeichen automatisch dann eine längere Pause zum Zeichen des Wort-Endes macht, wenn das im Bitmuster ganz rechts stehende Bit Nr. Ø den Wert 1 hat (in der Aufstellung grün eingetragenes Bit).

Wir wollen darauf verzichten, die Funktion des Morse-Programms ausführlich zu beschreiben. Wenn Sie diese Funktion interessiert, dann geben Ihnen die Kommentare in der Programm-Auflistung die Möglichkeit, das Programm in allen Einzelheiten zu analysieren. Wir wollen jedoch darauf hinweisen, daß in unserem Programm einige Routinen des Betriebsprogramms des Micro-Professors verwendet werden.

Das Programm soll nur soweit erläutert werden, wie es für das Verständnis der verwendeten Interrupt-Programmierung notwendig ist.

Sehen Sie sich bitte das Hauptprogramm ab der Adresse 18ØØH auf der Seite L 31 an! Weil wir für die Interrupt-Programmierung die PIO verwenden, wird zunächst die Initialisierungs-Routine INIT aufgerufen, die für die Programmierung der PIO zuständig ist und in der die CPU für Interrupts vorbereitet wird. Wir sehen uns diese Routine nachher noch genauer an.

Die Bytes für die Morsezeichen sind im Speicher ab dem Label TEXT bei der Adresse 18FØ eingetragen (Seite L 38). Im Anschluß an den Aufruf der INIT-Routine wird im HL-Registerpaar ein Zeiger auf das erste Text-Byte abgelegt.

In der beim Label BYTE beginnenden Programm-Schleife wird jeweils das Text-Byte in den Akkumulator geholt, auf das der Zeiger im HL-Registerpaar zeigt. Wenn dieses Byte den Wert ØØ hat, dann ist das ein Zeichen dafür, daß der gesamte Text als Morsezeichen ausgegeben worden ist; das Programm springt in diesem Fall zurück zum Label MORSE. Der Text-Zeiger im HL-Registerpaar wird dann wieder auf das erste Text-Zeichen bei der Adresse 18FØ gerichtet und das Spiel beginnt von neuem. (Überzeugen Sie sich davon, daß der Text im Speicher bei der Adresse 192C mit dem Byte ØØ abgeschlossen wird.)

Wenn im Hauptprogramm ein anderes als das Byte ØØ in den Akkumulator geholt worden ist, dann wird dieses Byte dem Programm-Modul KEY übergeben (CALL KEY). Hier wird das Byte mit dem Befehl SLA A Bit für Bit solange nach links aus dem Akkumulator hinausgeschoben, bis das in unserer Code-Auflistung rot eingetragene 1-Bit aus dem Akkumulator verschwunden ist. Je nach dem Wert des hinausgeschobenen Bits wird ein kurzes oder ein langes akustisches Zeichen generiert.

Am Ende der KEY-Routine wird der Inhalt des HL-Registerpaares inkrementiert. Der Zeiger zeigt jetzt auf den im Speicher nächstfolgenden Buchstaben. Das Hauptprogramm springt danach wieder zum Label BYTE und sorgt für die Ausgabe dieses nächstfolgenden Buchstabens.

Interrupts im Morse-Programm

Das soeben vorgestellte Hauptprogramm zur Ausgabe von Morsezeichen enthält keine Besonderheiten. Sein wichtigster Bestandteil ist die KEY-Routine, in der aus den im Speicher abgelegten Buchstaben-Bytes akustische Morsezeichen generiert werden. Im hier interessierenden Zusammenhang können wir diese Routine als *Black Box* betrachten: Als ein Ding, das die ihm übertragene Aufgabe erfüllt, ohne daß die interne Funktion untersucht werden muß.

Sehen (oder besser: hören) Sie sich zunächst einmal an, was das Morse-Programm tut.

Versuch S 34.1

Die Ausgabe von Morsezeichen

Tasten Sie bitte das ab der Seite L 31 aufgelistete Programm in Ihr System ein. Achten Sie dabei bitte darauf, daß auf der Seite L 38 im

Quellprogramm zwei ORG-Anweisungen stehen. Jede dieser ORG-Anweisungen veranlaßt den Assembler bei der Übersetzung, die anschließend folgenden Bytes ab der Adresse einzutragen, die in der ORG-Anweisung bestimmt worden ist (vgl. Seite L 2). Beim Eingeben unseres Programms müssen Sie also vor der Eingabe der zum Label VALARM gehörenden Bytes zunächst die Taste ADDR betätigen, die Adresse 18E8 eintasten und dann die Taste DATA betätigen. Erst dann dürfen Sie die Bytes 1ØH, 18H, 2FH und 18H eingeben.

Nach diesen vier Bytes folgt wieder ein Wechsel der Adressen, den der Assembler wegen der neuerlichen ORG-Anweisung berücksichtigt hat. Sie müssen also wieder mit Hilfe der ADDR-Taste die Adresse 18FØ vorwählen und können dann – nach Betätigung der DATA-Taste – die TEXT- und TEXT1-Bytes bis zur Adresse 1936H eingeben.

Das jetzt eingetastete Programm enthält außer dem eigentlichen Morse-Programm bereits alle Anweisungen zur Berücksichtigung von eventuellen Interrupts. Wir werden Ihnen gleich zeigen, welche Anweisungen das im einzelnen sind.

Zunächst können Sie das Programm einfach einmal bei der Adresse 18ØØH starten und sich anhören, was passiert. Es kommt gar nicht darauf an, daß Sie herausbringen, was da gemorst wird. Jedenfalls hört sich die Sache doch recht kommerziell an.

Wenn Sie wollen, dann können Sie den Ausgabe-Charakter der Zeichen versuchsweise ändern:

Eigenschaft	Adr.	Byte	Bemerkungen
Geschwindigkeit	18E5	1Ø – 7Ø	
Langes Zeichen-Element	18A7	Ø2 – Ø4	Ø3 ist normal
Zeichen/Pause	189C	Ø2 – Ø3	
Wort/Pause	188A	Ø4 – Ø6	
Tonhöhe	18EØ	8Ø – 6Ø	Geschw. beeinflußt

Zur Ausgabe der Morsezeichen wird die PIO in keiner Weise in Anpruch genommen. Der Aufruf der Initialisierungs-Routine (INIT) am Anfang des Programms ist also für die Ausgabe der Zeichen, die Sie sich jetzt angehört haben, nicht notwendig.

Die PIO wird in unserem Programm für die Verarbeitung von Interrupt-Anforderungen verwendet, und dazu muß sie in der INIT-Routine programmiert werden. Zusätzlich muß das Programm bei programmierter Interrupt-Möglichkeit natürlich zumindest eine Interrupt-Service-Routine enthalten.

Bei der Eingabe des Programms ist Ihnen sicher die gleich im Anschluß an das Hauptprogramm geschriebene ALARM-Routine aufgefallen, die mit einer RETI-Anweisung abschließt. Die ALARM-Routine ist also mit Sicherheit eine Interrupt-Service-Routine.

Sehen Sie sich die SOS-Routine auf der Seite L 33 an! Auch diese Routine schließt mit einer RETI-Anweisung ab. Sie haben es also offensichtlich mit einem Programm zu tun, das mit zwei unterschiedlichen Interrupts auch zwei unterschiedliche Interrupt-Service-Routinen aufrufen kann.

Ehe wir Ihnen zeigen, wie das funktioniert, sollen Sie sich im Versuch die Wirkung der Interrupts einmal ansehen.

S

36

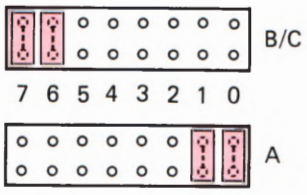


Bild S36.1
Für den Versuch S36.1 müssen die Stifte für die steckbaren Brücken so bestückt werden, wie es das Bild zeigt.

Versuch S36.1

Interrupts im Morse-Programm

Halten Sie das Programm bitte an und setzen Sie auf der Peripherie-Leiterplatte die steckbaren Brücken so auf die zu den Ports A und B gehörenden Stifte, wie es das Bild S36.1 zeigt. Starten Sie jetzt das Programm wieder.

Jetzt können wir in Gedanken ein Spielchen machen, das sicher nicht ganz praxisgerecht ist (Fachleute mögen uns das verzeihen!), das aber schön zu unserem Versuch paßt. – Stellen Sie sich vor, Sie säßen im Funkhaus eines Schiffes und hörten die Morsezeichen, die der Funker an seine Reederei schickt. (So ähnlich hört sich das wirklich an.)

Betätigen Sie die zum Port A gehörende Taste Nr. 0 auf der Peripherie-Leiterplatte! Die Funkerei wird unterbrochen, denn jetzt hat eine nicht zu überhörende Alarmklingel losgelegt, die einen katastrophalen Wassereinbruch aus dem Maschinenraum meldet. Der Funker reagiert blitzschnell: Er betätigt die zum Port B gehörende Taste Nr. 7. (Tun Sie das!).

Auch dann, wenn Ihnen Morsezeichen vollkommen fremd sind, werden Sie feststellen, daß der bisher eher unregelmäßige Zeichen-Rhythmus jetzt regelmäßiger geworden ist: Es werden drei kurze Zeichen ausgegeben, gefolgt von drei langen Zeichen und wieder drei kurzen Zeichen (dit dit dit, dah dah dah, dit dit dit, dit dit dit, usw). Dieser Rhythmus ist so charakteristisch, daß er zum internationalen See-Notzeichen geworden ist: SOS.

Die Betätigung der zum Port B gehörenden Taste Nr. 7 hat die normale Ausgabe der Morsezeichen unterbrochen und die dreimalige Aussendung der SOS-Folge ausgelöst. Nach der Ausgabe dieser Folge kehrt das Programm zur normalen Ausgabe von Morsezeichen zurück.

Unabhängig von der (zugegebenermaßen etwas fragwürdigen) Nutzanwendung unseres unterbrechbaren Morse-Programms können Sie die Alarm-Klingel oder wahlweise auch die SOS-Ausgabe zu beliebigen Zeiten der Morsezeichen-Ausgabe wiederholen. Die Sache wird jedesmal funktionieren. Schon daran erkennen Sie nach Ihren bisherigen Erfahrungen, daß es sich bei den ALARM- und SOS-Routinen um echte Interrupt-Service-Routinen handelt.

Die Programmierung einer einzelnen dieser Routinen darf Ihnen kaum mehr wie Hexenwerk vorkommen. Überlegen Sie zum Beispiel, wie der Interrupt für die Interrupt-Service-Routine ALARM programmiert werden muß:

Die Anfangsadresse der Routine wird im Speicher in zwei aufeinanderfolgenden Speicherzellen abgelegt: In der Speicherzelle mit der niedrigeren Adresse wird das LOB dieser Anfangsadresse abgelegt und in der Speicherzelle mit der höheren Adresse das HOB. Bedingung dabei ist, daß die Ablage-Adresse für das LOB ohne Rest durch zwei teilbar ist (vgl. Seite S24). – Wir haben die Ablage-Adresse für das LOB der ALARM-Startadresse in unserem Programm mit VALARM bezeichnet.

Weil der Interrupt über eine zum PIO-Port A gehörende Taste ausgelöst werden soll, muß dieser PIO-Port über den System-Port

PIOBA (Port-Adresse 82H) entsprechend der Zusammenstellung auf der Seite T 3 für die Betriebsart 3 programmiert werden (Byte **CF**).

Der Interrupt soll wahlweise über die Taste Nr. 0 **ODER** über die Taste Nr. 1 ausgelöst werden können. Nach dem *Mode Control Word* CF für die Betriebsart 3 muß das *I/O Register Control Word* mit dem Byte **Ø3** die entsprechenden Port-Anschlüsse als Eingänge definieren.

Anschließend wird der PIO mit dem *Interrupt Vector Word* das LOB für den Interrupt-Vektor übergeben, der auf die Speicherzelle VALARM zeigt. Im Quellprogramm bezeichnen wir dieses LOB als **LOW VALARM.** Das HOB der Speicherzelle VALARM (HIGH VALARM) wird später in das I-Register der CPU eingetragen.

Das jetzt folgende *Interrupt Control Word* bestimmt die Interrupt-Bedingungen. Der Wert 1 des Bits Nr. 4 kündigt der PIO an, daß noch eine Interrupt-Maske folgt. – Der Interrupt soll durch die Betätigung einer Taste, also mit einem Ø-1-Übergang des Interrupt-Signals erfolgen. Das Bit Nr. 5 muß deshalb des Wert Ø haben. – Die beiden Interrupt-auslösenden Tasten werden ODER-verknüpft. Das wird durch den Wert Ø des Bits Nr. 6 eingestellt. – Schließlich wird die PIO mit dem Wert 1 des Bits Nr. 7 zur Weitergabe einer Interrupt-Anforderung ermächtigt. – Mit den Werten Ø111 der Bits Nr. 0 bis Nr. 3 ergibt sich damit für das *Interrupt Control Word* das Bitmuster 1Ø11 Ø111 (Byte **B7**).

Als letztes wird dem PIO-Port A das *Mask Control Word* zugesandt, in dem die beiden vorher als Eingänge definierten Port-Anschlüsse zu Interrupt-Eingängen gemacht werden. Dazu werden die beiden Bits Nr. 0 und Nr. 1 auf die Werte Ø gesetzt; alle anderen Bits erhalten die Werte 1. (Byte **FC**).

Vergleichen Sie bitte die hier entworfene Programmierung des PIO-Ports A mit den ersten zehn Befehlen der Initialisierungs-Routine INIT auf der Seite L 34! Sie erkennen, daß das Programm genau entsprechend ausgelegt worden ist.

Das ist soweit in Ordnung: Die Interrupt-Programmierung für die ALARM-Routine bietet keinerlei Besonderheiten. Neu ist aber, daß unser Programm nicht nur eine, sondern zwei verschiedene Interrupt-Routinen aufrufen kann.

Sehen Sie sich bitte das Bild S 38.1 auf der folgenden Seite an! Es ist gerade so angelegt wie das Bild S 22.1 und zeigt links die Belegung des Speichers mit dem Morse-Programm. Mit blauer Unterlegung haben wir die Interrupt-Service-Routine ALARM gekennzeichnet. Die Anfangsadresse dieser Routine ist in den Speicherzellen mit den Adressen 18E8 und 18E9 abgelegt; der Interrupt-Vektor des PIO-Ports A zeigt auf die erste dieser beiden Speicherzellen.

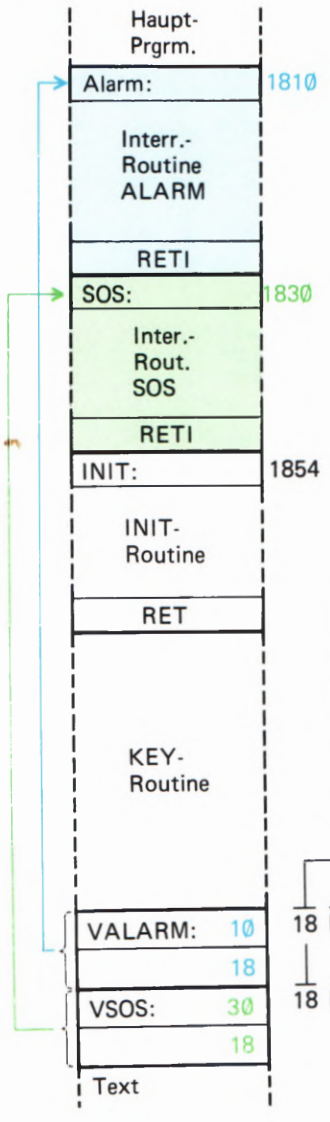
Wenn Sie jetzt die Interrupt-Service-Routine ALARM einmal ganz vergessen und nur die Interrupt-Service-Routine SOS ansehen, dann stellen Sie fest, daß diese Routine doch im Prinzip ganz genau so programmiert wird, wie Sie es bereits kennengelernt haben: Die Anfangsadresse der Routine wird (ganz egal, wo sie im Speicher angeordnet ist) in zwei aufeinanderfolgenden Speicherzellen abgelegt. Es ist wiederum egal, wo im Speicher diese beiden Ablage-Speicherzellen angeordnet sind. Einzige Bedingung ist zunächst nur, daß die Adresse der ersten dieser beiden Speicherzellen ohne Rest durch zwei teilbar ist.

S
38

Wenn die Interrupt-Service-Routine SOS über den PIO-Port B angesprochen werden soll, dann muß das LOB der Adresse der ersten der beiden Ablage-Speicherzellen dem PIO-Port B in der Initialisierungs-Routine als *Interrupt Vector Word* übergeben werden. Das HOB dieser Adresse muß im I-Register der CPU stehen.

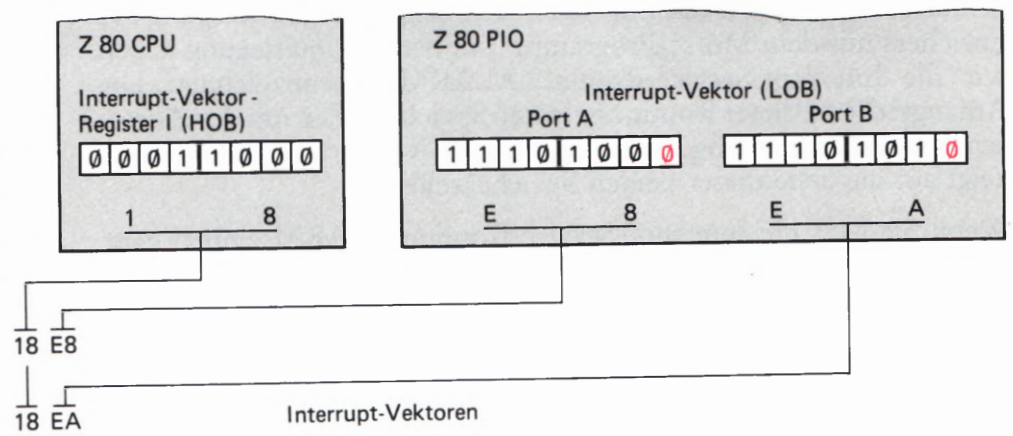
In unserem Programm folgen die beiden Interrupt-Service-Routinen ALARM und SOS mehr oder weniger zufällig im Speicher unmittelbar nacheinander. Wir haben das wegen der Übersichtlichkeit des Programms so eingerichtet. – Daß die Ablage-Speicherzellen für die Anfangsadressen der Interrupt-Service-Routinen im Speicher unmittelbar aufeinander folgen, ist kein Zufall. Die HOBs ihrer Adressen müssen zwangsweise gleich sein, denn dieses HOB ist ein für alle mal im I-Register der CPU abgelegt. Und es ist durchaus sinnvoll und naheliegend, die Anfangsadressen aller in einem Programm verwendeten Interrupt-Service-Routinen – wo immer diese auch liegen mögen – einfach in einem **Vektorfeld** im Speicher hintereinander abzulegen. In der ersten, dritten usw. Speicherzelle dieses Vektorfelds steht jeweils das LOB einer Anfangsadresse, und wenn Sie die Sache so betrachten, dann wird es auf einmal sehr sinnvoll, daß die Bitmuster der Adressen-LOBs der Ablage-Speicherzellen am Ende jeweils ein Ø-Bit enthalten müssen.

Bild S 38.1
Bei verschachtelten Interrupts sind die Anfangsadressen der Interrupt-Service-Routinen in einem Vektorfeld angeordnet.



Die Anlage eines solchen Vektorfeldes macht es sehr einfach, dem Port eines bestimmten Peripherie-Bausteins eine beliebige der programmierten Interrupt-Service-Routinen zuzuordnen: Man braucht ihm nur ein *Interrupt Vector Word* zu übermitteln, in dem das LOB der betreffenden Adresse im Vektorfeld enthalten ist.

Sehen Sie sich jetzt die Programmierung des PIO-Ports B in der INIT-Routine ab der Seite L 34 an! Die übermittelten Bytes bedürfen kaum einer Erläuterung. Auch dieser Port wird für die Betriebsart 3 programmiert. Im *I/O Register Control Word* und im *Mask Control Word* wird dafür gesorgt, daß ein Interrupt über die Tasten Nr. 6 und Nr. 7 ausgelöst werden kann. Das *Interrupt Control Word* ist das gleiche wie das für den Port A. Der wichtigste Unterschied besteht in den *Interrupt Vector Words* für die beiden Ports. Das Byte LOW VSOS (= EA) ist genau um zwei größer als das Byte LOW VALARM (= E8).



Die restlichen Befehle der INIT-Routine sind Ihnen geläufig. Zuerst wird mit zwei Befehlen das HOB des Vektorfelds in das I-Register der CPU eingetragen. Anschließend wird die CPU mit dem Befehl IM 2 für den Interrupt-Mode 2 programmiert und dann wird das Interrupt-Flipflop IFF1 mit dem Befehl EI gesetzt, damit die CPU Interrupts überhaupt verarbeiten kann.

Versuch S 39.1

Interrupt-Priorität

Lassen Sie unser Schiffsuntergang-Spielchen jetzt einmal beiseite und betätigen Sie bei laufender Morsezeichen-Ausgabe die SOS-Taste (Port B, Nr. 7). Betätigen Sie sofort anschließend, wenn die Ausgabe der SOS-Zeichen begonnen hat, die ALARM-Taste (Port A, Nr. 0). Was geschieht?

Zunächst geschieht gar nichts. Aber dann, wenn die Ausgabe der drei SOS-Folgen abgelaufen ist und Sie die ALARM-Taste längst wieder losgelassen haben, ertönt die Alarm-Klingel. Sie stellen also zweierlei fest: Erstens ließ sich die laufende Interrupt-Service-Routine für den SOS-Ruf nicht durch den ALARM-Interrupt unterbrechen, und zweitens hat sich die PIO die ALARM-Interrupt-Anforderung gemerkt.

Daß sich die SOS-Ausgabe nicht unterbrechen ließ, liegt einfach daran, daß in der CPU nach dem angenommenen ersten Interrupt das Interrupt-Flipflop IFF1 zurückgesetzt wurde. Es wird erst am Ende der SOS-Routine mit dem Befehl EI (Adresse 1852H) wieder gesetzt.

Durch die ALARM-Interrupt-Anforderung entstand dann ein *Pending Interrupt*, ein „hängender" Interrupt, der angefordert, aber noch nicht ausgeführt wurde.

Könnte man erreichen, daß die SOS-Routine ihrerseits von einer Interrupt-Anforderung unterbrochen werden kann? Natürlich könnte man! Man braucht nur dafür zu sorgen, daß das Interrupt-Flipflop IFF1 in der CPU gleich nach dem Aufruf der SOS-Routine wieder gesetzt wird (vgl. Seite H 40). Wir haben dafür vorsorglich am Anfang der SOS-Routine bei der Adresse 183ØH einen NOP-Befehl programmiert.

Halten Sie das Programm an und ersetzen Sie den NOP-Befehl bei der Adresse 183ØH durch einen EI-Befehl (Byte FB)! Den EI-Befehl am Ende der SOS-Routine (Adresse 1852H) brauchen Sie für diesen Versuch nicht zu löschen. Er versucht nur, das bereits gesetzte Interrupt-Flipflop IFF1 neuerlich zu setzen, was sicher nicht schadet.

Starten Sie das Programm und wiederholen Sie den eben beschriebenen Versuch! Sie stellen fest, daß die Interrupt-Service-Routine SOS jetzt ihrerseits durch einen ALARM-Interrupt unterbrochen werden kann. Nach dem Ablauf dieses Alarms wird die SOS-Routine bis zu deren Ende fortgesetzt.

Jetzt wäre es interessant festzustellen, ob auch die Interrupt-Service-Routine ALARM durch einen SOS-Interrupt unterbrochen werden

kann. Ersetzen Sie den für einen entsprechenden Versuch vorsorglich programmierten NOP-Befehl am Anfang der ALARM-Routine (Adresse 181ØH) durch das Byte FB für einen EI-Befehl! Versuchen Sie jetzt, sofort im Anschluß an einen ALARM-Interrupt, solange die Klingel noch in Tätigkeit ist, einen SOS-Interrupt auszulösen!

Der Versuch gelingt nicht. Der ALARM-Interrupt wird zunächst vollständig erledigt, und erst dann wird der *pending* SOS-Interrupt bedient. Und das, obwohl in der CPU das Interrupt-Flipflop IFF1 wieder gesetzt und die CPU bereit ist, sofort einen weiteren Interrupt auszuführen. Wie kommt das?

Offensichtlich wird die an den PIO-Port B gelieferte Interrupt-Anforderung nicht an die CPU weitergereicht. Richtig! Geradeso wie in der *Daisy Chain* verschiedener Peripherie-Bausteine besteht auch innerhalb des Peripherie-Bausteins PIO eine Interrupt-Hierarchie. Der PIO-Port A hat eine höhere Interrupt-Priorität als der PIO-Port B. Ein über den PIO-Port B angeforderter Interrupt wird erst dann an die CPU weitergeleitet, wenn dem PIO-Port A durch einen RETI-Befehl mitgeteilt worden ist, daß die von ihm angeforderte Bearbeitung seiner Interrupt-Service-Routine abgeschlossen ist.

Wir kommen auf eine solche interne Interrupt-Hierarchie bei der Vorstellung des CTC-Bausteins noch einmal zurück.

Der Z 80 CTC

Bei der Vorstellung dieses Bauelements geraten wir gleich zu Anfang in Schwierigkeiten: Wie sollen wir es bezeichnen? – Hier macht sich die amerikanische Abstammung besonders bemerkbar. Unter Fachleuten ist der Begriff PIO inzwischen weitgehend eingeführt; PIOs gibt es in fast allen Mikroprozessor-Familien. Die Bezeichnung CTC wird Ihnen weitaus seltener begegnen.

CTC ist eine der im englischen Sprachgebrauch so sehr beliebten Abkürzungen und kommt von *Counter/Timer Circuit.* Wörtlich übersetzt bedeutet das: Zähler/Zeitgeber Schaltkreis.

Wir werden – ähnlich wie bei der PIO – bei der Bezeichnung CTC bleiben und dabei dem Bauelement mehr oder weniger willkürlich männliche Eigenschaften zuordnen: **Der** CTC, weil es **der** Schaltkreis heißt.

Was macht man mit dem CTC?

Die in unserem Lehrgang interessierenden Peripherie-Bausteine haben vorwiegend die Eigenschaften von Interface- (Schnittstellen-)Bausteinen (vgl. Seite H7). Solche Bausteine vermitteln den Datenverkehr zwischen einem Mikroprozessor-System und seiner Peripherie. Der CTC nimmt in diesem Zusammenhang ein wenig eine Sonderstellung ein. Er kann zwar auch eine Vermittler-Rolle zwischen Peripherie und System übernehmen, eignet sich aber mindestens genauso dazu, selbst Peripherie zu spielen.

Sehen Sie sich bitte das Bild S41.1 an! In Anlehnung an die Darstellung im Bild H7.1 wird ein recht einfaches Mikroprozessor-System mit der CPU, dem Speicher und einer PIO als Schnittstellen-Baustein gezeigt. Die PIO vermittelt den Datenverkehr zwischen dem System und einer irgendwie gearteten Peripherie. Diese Peripherie soll in regelmäßigen zeitlichen Abständen veranlaßt werden, Daten an das System zu liefern. Die Länge dieser zeitlichen Abstände bestimmt das System. Hier liegt das einfachste Beispiel für die Verwendung eines CTC in seiner Eigenschaft als Zeitgeber (Timer) vor.

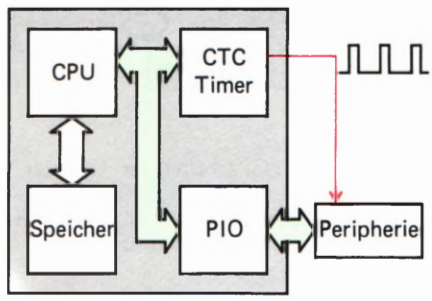


Bild S41.1
Der Z 80 CTC kann in der Timer-Betriebsart regelmäßige Impulse an die Peripherie liefern.

Das Bild S41.1 zeigt, daß der CTC als Peripherie-Baustein offenbar genauso in das System eingefügt wird wie der Peripherie-Baustein PIO. (Daß er tatsächlich genauso eingefügt wird, werden Sie in einem der folgenden Abschnitte noch sehen.) Das Bild zeigt aber auch den ganz wesentlichen Unterschied zwischen den Peripherie-Bausteinen PIO und CTC: Die PIO wickelt den Datenverkehr mit der Peripherie Bit-parallel, Byte-seriell (vgl. Seite H3) über acht Daten-Anschlüsse ab. Im Gegensatz dazu verfügt der im Bild dargestellte Timer nur über einen einzigen Ausgangs-Anschluß, über den der Peripherie Daten geschickt werden können.

Sie müssen es zunächst einfach glauben, daß der Ausgangs-Anschluß des Timers keineswegs dazu geeignet ist, Daten Bit-seriell, Byte-seriell zu übermitteln, wie man vielleicht vermuten möchte. Eine solche

Datenübertragung ist nicht Aufgabe des Timers. Seine Funktion ist es vielmehr, in regelmäßigen Abständen – sozusagen als Uhr – an die Peripherie Impulse zu liefern. Und das ist genau die Aufgabe, die wir dem CTC im angegebenen Beispiel stellen.

Wie die Peripherie aufgebaut sein muß, daß sie jedesmal beim Eintreffen eines Timer-Impulses Daten an das System liefert, ist hier ganz gleichgültig. Viel interessanter ist es, wie die zeitlichen Abstände der vom Timer gelieferten Impulse eingestellt werden können.

Unser Bild zeigt, daß der Timer – wie auch die PIO – an den (grün gekennzeichneten) Datenbus des Systems angeschlossen ist. Bitte erinnern Sie sich: Der Anschluß der Peripherie an den Datenbus des Systems hat zwei Gründe. Zum einen werden der PIO über den Datenbus die an die Peripherie zu übermittelnden Bytes übergeben. Zum andern werden der PIO aber über den Datenbus auch Befehls-Bytes geschickt, mit denen ihr Verhalten programmiert werden kann. Nur aus diesem zweiten Grund ist auch der CTC an den Datenbus des Systems angeschlossen: Er kann programmiert werden.

Die Programmierung des CTC soll uns nachher noch ausführlich beschäftigen. Sie ist entscheidend wichtig und kann den Timer zu einer Reihe unterschiedlicher Tätigkeiten veranlassen. Für das im Bild S 41.1 dargestellte Beispiel muß der Timer zunächst auf die Ausgabe von Impulsen programmiert werden. Eine zusätzliche Programmierung erlaubt es darüber hinaus, auch die Frequenz der ausgegebenen Impulse, also den zeitlichen Abstand der Impulse untereinander, einzustellen.

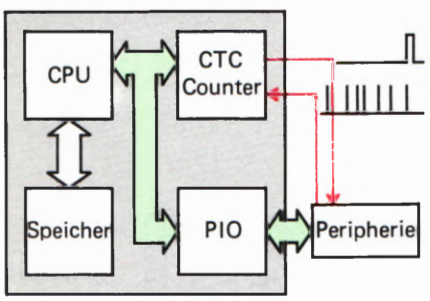


Bild S 42.1
Der CTC arbeitet im Counter-(Zähler-)Mode. Nach einer voreinstellbaren Anzahl von Impulsen sendet er ein Signal an die Peripherie.

Im Bild S 42.1 haben wir eine andere der vielen Anwendungsmöglichkeiten des CTC angedeutet. Zunächst wird Ihnen im Vergleich zum vorhergehenden Bild auffallen, daß diesmal zwei Anschlüsse für Verbindungen zur Peripherie dargestellt sind: Der bereits bekannte Impuls-Ausgang und zusätzlich ein Eingang, über den die Peripherie Impulse an den CTC schicken kann. In das Bild ist eine Folge unregelmäßig eintreffender, sehr kurzer Impulse von der Peripherie eingetragen.

Der CTC hat in diesem Fall eine ganz andere Aufgabe als im ersten Beispiel. Die Peripherie meldet irgendwelche internen Ereignisse jeweils durch einen kurzen Impuls an den CTC. Dort werden die Impulse gezählt, und nach einer voreinstellbaren Anzahl dieser Ereignisse sendet der CTC seinerseits einen einzelnen Impuls an die Peripherie.

Der CTC arbeitet jetzt als voreinstellbarer Ereignis-Zähler (Counter). Auch diese Art der Tätigkeit wird mit Initialisierungs-Befehlen über den Datenbus programmiert.

Eine ganz ähnliche Aufgabe ist im Bild S 43.1 angedeutet. Auch jetzt treffen von der Peripherie zu unregelmäßigen Zeitpunkten Impulse ein, die in dem als Counter arbeitenden CTC gezählt werden. Wenn die voreingestellte Anzahl von Impulsen eingetroffen ist, dann wird das diesmal nicht mit einem Impuls an die Peripherie gemeldet, sondern an die CPU. Die CPU reagiert auf diese Meldung durch den Aufruf eines bestimmten Unterprogramms, in dem z. B. die Aussendung eines oder mehrerer Bytes über die PIO an die Peripherie programmiert ist.

Wir haben ausdrücklich darauf hingewiesen, daß die Impulse von der Peripherie in unregelmäßigen Abständen beim CTC eintreffen. Es ist also auch nicht vorhersagbar, wann die voreingestellte Anzahl dieser Impulse erreicht ist. Weil das zugehörige Unterprogramm jedoch sofort nach Erreichen der vorgegebenen Impuls-Zahl mit seiner Arbeit beginnen soll, liegt es nahe, das Unterprogramm als Interrupt-Service-Routine aufzurufen.

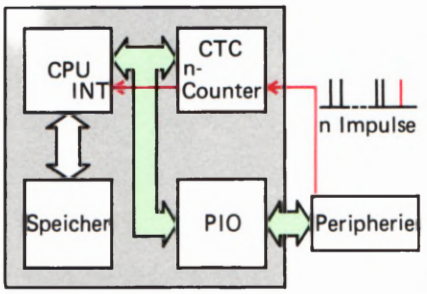


Bild S43.1
Der CTC zählt von der Peripherie gelieferte Impulse. Nach einer voreinstellbaren Anzahl von Impulsen übermittelt er eine Interrupt-Anforderung an die CPU.

Hier lernen Sie einen dritten Anschluß des CTC kennen, der aber nicht zur Peripherie führt, sondern innerhalb des Systems verwendet wird. Es ist der Interrupt-Anschluß $\overline{INT}$, der System-intern mit dem entsprechenden Anschluß der CPU verbunden wird.

Es ist Aufgabe der Programmierung des CTC, die Tatsache des Erreichens einer vorgegebenen Anzahl von Zähl-Impulsen entweder über den Peripherie-Anschluß in Form eines Impulses auszugeben, oder diese Tatsache System-intern, ebenfalls mit einem entsprechenden Signal, der CPU als Interrupt-Anforderung zu übermitteln.

Es soll bereits an dieser Stelle darauf aufmerksam gemacht werden, daß die Voreinstellung des CTC durch das Füllen eines internen Zählers im CTC mit dem geforderten Wert vorgenommen wird. Jeder Zähl-Impuls dekrementiert diesen internen Zähler; wenn der Zähler bis auf den Wert Ø leergezählt ist, erfolgt die Meldung entweder an die Peripherie oder über den Interrupt-Anschluß. Den Zähler-Mechanismus stellen wir Ihnen nachher noch genauer vor.

Ein letztes Beispiel dieser Art zeigt das Bild S43.2. Auch hier meldet die Peripherie interne Ereignisse mit Impulsen an den CTC. Diesmal werden aber die Peripherie-Impulse nicht abgezählt. Jeder Impuls wird eigens durch einen vom CTC ausgesandten Impuls quittiert. Das Quittungs-Signal erscheint aber mit einer voreinstellbaren Verzögerung.

Der CTC arbeitet nicht als Zähler (Counter), sondern als eine Art Zeitgeber (Timer). Die Sache funktioniert so: Beim Eintreffen eines Impulses von der Peripherie ist der CTC-interne Zähler mit einem bestimmten (durch die Programmierung einstellbaren) Wert gefüllt. Der Impuls startet das Leer-Zählen des Zählers in einem Rhythmus, der vom CPU-Takt bestimmt wird. Je höher der Wert ist, mit dem der Zähler vorab gefüllt ist, um so länger dauert es, bis der Zähler leergezählt ist, um so länger ist also die Verzögerungszeit des Quittungs-Signals.

In den hier vorgestellten Beispielen arbeitet der CTC eindeutig als Interface-Baustein. Er überträgt zwar keine Daten, wie sie z.B. als Bytes im Speicher des Systems abgelegt sind, aber auch einzelne Impulse können ja durch ihre zeitliche Lage Informationen enthalten.

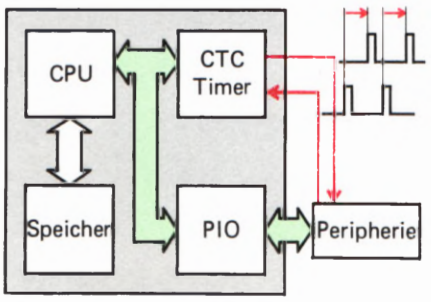


Bild S43.2
Bei spezieller Programmierung kann der CTC nach einer vorgegebenen Zeit an die Peripherie das Echo eines Impulses liefern.

Am Anfang dieses Abschnitts haben wir behauptet, daß der CTC auch selbst Peripherie spielen kann. Wir werden gerade von dieser Fähigkeit in einigen Programmen Gebrauch machen. Hier soll nur kurz erläutert werden, was es damit auf sich hat.

Nehmen Sie an, Sie wollten Ihren Micro-Professor so programmieren, daß er in seiner Sieben-Segment-Anzeige die aktuelle Uhrzeit anzeigt. Wie ein solches Programm im Einzelnen aussieht, braucht hier nicht

zu interessieren. (Im Lehrgang Mikroprozessortechnik ist im Bild H9.1 ein Beispiel aufgelistet.) Im einfachsten Fall wird so programmiert, daß das Programm in einer Schleife läuft, die in genau einer Sekunde abgearbeitet wird. Innerhalb dieser Schleife werden die Anzeige-Bytes aufgearbeitet, das Sekunden-Byte inkrementiert und die Anzeige bedient. Wenn diese Aufgabe von der CPU jeweils in kürzerer Zeit als in einer Sekunde erledigt wird (was sicher der Fall ist), dann wird die Bearbeitungszeit der Schleife durch eine interne Warte-Schleife entsprechend verlängert.

Sie wissen inzwischen, daß ein solches Verfahren eigentlich recht unökonomisch ist, weil Warte-Schleifen im Grunde verschwendete Zeit darstellen, in der die CPU viel nützlichere Arbeit erledigen könnte. Viel effektiver wäre es, wenn nach Ablauf einer jeden Sekunde ein Interrupt erfolgte. In der dann aufgerufenen Interrupt-Service-Routine könnten alle Arbeiten zur Anzeige der Uhrzeit erledigt werden, und dann könnte sich die CPU wieder ihrer eigentlichen Aufgabe widmen. Bei einer solchen Programmierung kann die Uhr sozusagen nebenbei laufen, ohne daß die CPU sonderlich in Anspruch genommen wird.

Gut und schön, aber wo sollen immer nach Ablauf einer Sekunde die Interrupt-Anforderungen herkommen? Man kann natürlich einen peripheren Quarz-Oszillator bauen, der z. B. über einen PIO-Port jede Sekunde eine Interrupt-Anforderung an die CPU schickt.

Genau das ist nicht notwendig, wenn man einen CTC zur Verfügung hat. Dieser CTC kann so programmiert werden, daß ein eingebauter Zähler genau nach Ablauf einer Sekunde leergezählt ist. Jedesmal dann, wenn das der Fall ist, generiert der CTC ein Interrupt-Signal für die CPU. Der CTC ersetzt also den peripheren Quarz-Oszillator und spielt selbst Peripherie.

Wir haben Ihnen hier nur einige Beispiele für den sinnvollen Einsatz des CTC gezeigt. Entscheidend für die Anwendung des CTC ist seine Programmierung, und die sollen Sie im folgenden Abschnitt kennenlernen.

Die Programmierung des CTC

Das Prinzip der Programmierung eines Peripherie-Bausteins haben Sie bereits im Zusammenhang mit der Z 80 PIO kennengelernt. Um es vorweg zu sagen: Der CTC ist viel einfacher zu programmieren als die PIO.

Der PIO werden über den Datenbus des Systems zwei ganz unterschiedliche Arten von Bytes übergeben: Solche, die sie letztlich an die Peripherie weitergeben muß, und solche, die zu ihrer Programmierung dienen. Aus diesem Grunde gehören auch zu jedem der beiden PIO-Ports zwei (Port-)Adressen des Systems: Über die eine Adresse werden der PIO die Befehls-Bytes zugesandt (PIOBA, PIOBB), über die andere Adresse erhält die PIO die Daten, die sie an die Peripherie weiterleiten muß (PIODA, PIODB).

Diese Unterscheidung braucht beim CTC nicht getroffen zu werden. Der CTC gibt an die Peripherie keine acht Bit „breiten" Daten-Bytes von der CPU weiter, sondern nur einzelne Bits in Form von Impulsen. Die Impulse werden vom CTC selbst generiert, und zwar immer dann, wenn ein interner Zähler leergezählt worden ist. Der CTC kennt also nur eine Befehlsadresse, über die er programmiert wird, und damit hat sich's. Über diese Befehlsadresse werden ihm nacheinander die Befehls-Bytes übergeben, die – ähnlich wie bei der PIO – durch Kennungen unterschieden werden.

Das Elegante bei der Sache ist, daß der CTC nur zwei verschiedene Arten von Befehlen braucht: Ein *Channel Control Word*, das dem *Interrupt Control Word* der PIO entspricht, und ein *Interrupt Vector Word*, das die gleiche Funktion wie das bei der PIO hat.

Wie das *Interrupt Control Word* bei der PIO, so kann auch das *Channel Control Word* des CTC als Ein- oder Zwei-Byte-Befehl auftreten. Abhängig vom Wert eines bestimmten Bits erwartet der CTC nach dem ersten Byte des *Channel Control Words* ein zweites Byte mit zusätzlichen Informationen.

Eigentlich ist dieses zweite Byte des *Channel Control Words* sogar das wichtigste bei der ganzen Angelegenheit, denn es enthält die Information, mit welchem Wert der interne Zähler im CTC geladen werden soll. Und damit wird die Zeit bestimmt, die vom Beginn des Leer-Zählens des Zählers bis zu seinem Inhalt Ø vergeht. Sie erinnern sich: Beim Inhalt Ø des Zählers wird ein Impuls an die Peripherie abgegeben bzw. eine Interrupt-Anforderung an die CPU geschickt.

Wegen dieser Wichtigkeit hat das zweite Byte des *Channel Control Words* einen eigenen Namen. Es heißt *Time Constant Word*, was einfach mit Zeitkonstante übersetzt werden kann.

Im Bild S 45.1 sind die Befehls-Bytes für den CTC (ähnlich wie die Befehls-Bytes für die PIO auf der Seite T 3) zusammengestellt. Wir haben auch gleich die Bedeutung der einzelnen Bits eingetragen, auf die wir nachher eingehen.

Zunächst soll nur festgestellt werden, daß bei der erstmaligen Programmierung des CTC in der Initialisierungs-Routine (INIT) das *Channel Control Word* immer ein Zwei-Byte-Befehl sein muß. Mit anderen Worten: Dem *Channel Control Word* muß ein *Time Constant Word* folgen. Aus den in das Bild eingetragenen Bedeutungen der Bits sehen Sie schon jetzt, daß dann das Bit Nr. 2 im *Channel Control Word* den Wert 1 haben muß.

Ehe wir Ihnen die Bedeutungen der einzelnen Bits in den Befehls-Bytes erläutern, soll noch eine wichtige Eigenschaft des CTC nachgetragen werden, auf die Sie vielleicht schon bei der Betrachtung des *Interrupt Control Words* aufmerksam geworden sind. Und dann sollen Sie erst einmal einen Versuch mit dem CTC machen.

Zunächst aber unser Nachtrag: In den Bildern zu den Verwendungs-Beispielen des CTC (Bilder S 41.1 bis S 43.2) ist der CTC jeweils so dargestellt, als enthalte er intern nur einen einzigen Zähler, der entweder als reiner Zähler (Counter) arbeitet, oder – bei entsprechender Programmierung – als Zeitgeber (Timer). In Wirklichkeit ist der CTC

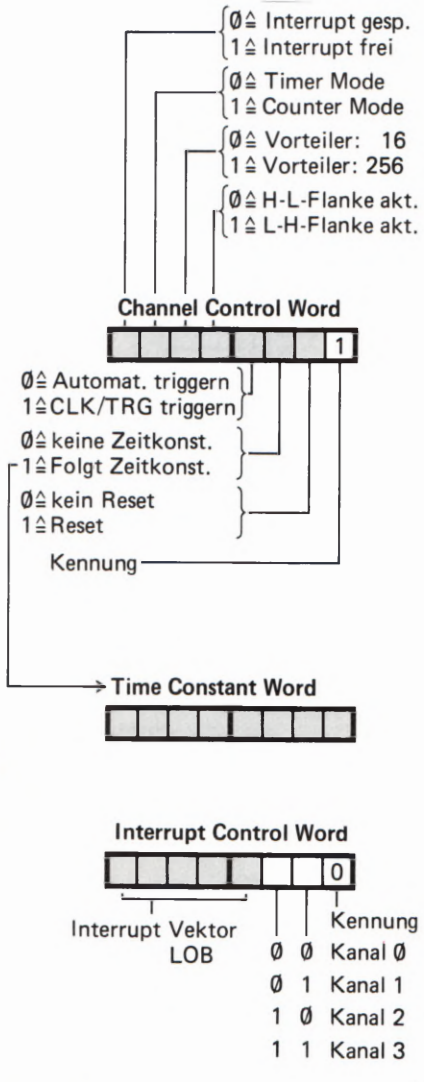


Bild S 45.1
Jeder CTC-Kanal wird mit einem Channel Control Word und mit einem Time Constant Word programmiert. Das Interrupt Vector Word hat für alle vier Kanäle Gültigkeit.

noch ein bißchen komfortabler: Das Bauelement enthält insgesamt vier solcher Zähler und dementsprechend auch (mit einer Ausnahme) vier mal die in den Bildern dargestellten Ein- und Ausgangs-Anschlüsse. In der Praxis könnte ein einziger CTC-Baustein also alle vier der angeführten Beispiele gleichzeitig erledigen.

Jeder der Zähler kann (mit allen damit verbundenen Möglichkeiten) einzeln programmiert werden. Bei der Programmierung werden die Zähler als Kanal 0 bis Kanal 3 (*Channel* 0 bis *Channel* 3, sprich: tschännel) bezeichnet.

Hier liegt der Grund, daß dem CTC im System nicht nur eine einzige Programmierungs- (Port-)Adresse zugeordnet ist, sondern daß jeder Kanal eine eigene Programmierungs- (Port)-Adresse hat. Im Micro-Professor hat der Kanal 0 die Adresse 40H und der Kanal 3 die Adresse 43H.

Es muß noch angemerkt werden, daß die vier Kanäle des CTC nicht völlig gleichwertig sind. Sehen Sie sich bitte das Bild S 46.1 an! Es ist das für die Programmierung wichtige Prinzip des CTC dargestellt. Sie erkennen, daß jeder der Kanäle 0 bis 3 über einen Eingang (CLK/TRG) und über zwei Ausgänge (ZC/TO und INT) verfügt. Eine Ausnahme macht der Kanal 3, bei dem der Ausgang ZC/TO fehlt. Für einen solchen Ausgang stand am Gehäuse des ICs einfach kein Anschluß mehr zur Verfügung.

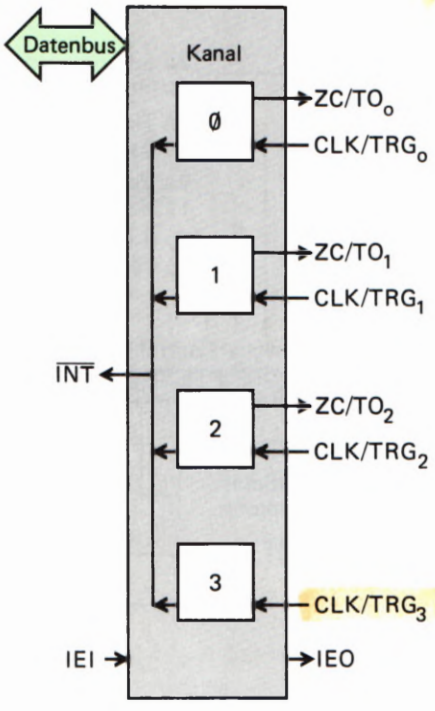


Bild S 46.1
Jeder der vier CTC-Kanäle hat einen CLK/TRG-Eingang. Bei den ersten drei CTC-Kanälen ist außerdem ein ZC/TO-Ausgang zugänglich.

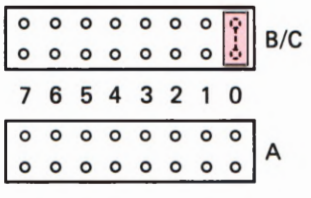


Bild S 46.2
Für den Versuch S 46.1 muß die Stiftleiste für den PIO-Port B beim Bit 0 mit einer steckbaren Brücke bestückt werden.

Versuch S 46.1

Der CTC als Ereignis-Zähler

Schließen Sie die Peripherie-Leiterplatte an Ihr System an und verbinden Sie den auf der Leiterplatte mit 0 gekennzeichneten Anschluß der Klemmleiste über ein kurzes Stück Draht mit dem Anschluß CLK/TRG0 der Klemmleiste. – Setzen Sie eine steckbare Brücke auf die mit 0 bezeichneten Stifte für den Port B (Bild 46.2).

Tasten Sie das auf den Seiten L 45 und L 46 aufgelistete Programm in Ihr System ein und beachten Sie, daß die Ablage Speicherzellen für die Anfangsadresse der Interrupt-Service-Routine die Adressen 1830 und 1831H haben.

Starten Sie das Programm! – Abgesehen davon, daß die Anzeige des Micro-Professers verlöscht, geschieht zunächst gar nichts.

Betätigen Sie jetzt in beliebigen zeitlichen Abständen mehrfach die Taste Nr. 0 des PIO-Ports B! Sehen Sie zu, was passiert! – Wiederholen Sie die Eingaben und versuchen Sie, das Prinzip des Geschehens zu erkennen.

Der Versuch realisiert das im Bild S 43.1 dargestellte Beispiel. Über die steckbare Brücke Nr. 0 wird ein mit der Taste Nr. 0 des PIO-Ports B eingegebenes 1-Signal auf den Anschluß 0 der Klemmleiste geschaltet (vgl. das Bild H 12.1). Dieses Signal ist aber in unserem Versuch nicht für die PIO bestimmt; es wird vielmehr über die Draht-Verbindung an den Eingang CLK/TRG0 des CTC-Kanals 0 geschaltet.

Im Programm besorgt die INIT-Routine die Initialisierung des Kanals 0 des CTC. Das *Channel Control Word* sorgt dafür, daß der Kanal 0

des CTC als Zähler für Impulse arbeitet, die über den Anschluß CLK/TRG0 eingegeben werden. Der zu diesem Kanal gehörende Zähler wird mit dem anschließenden *Time Constant Word* auf den Wert 5 geladen.

Jeder über den Anschluß CLK/TRG0 eintreffende Impuls dekrementiert den Zähler. Nach dem fünften Impuls ist der Zähler leergezählt: Sein Inhalt ist 0. Diese Tatsache veranlaßt den CTC, einen Impuls über den Ausgang ZC/TO0 auszugeben. Es sei hier schon verraten, daß die Abkürzung ZC vom englischen *Zero Count* kommt, und das bedeutet: Auf null gezählt.

Das Signal am Ausgang ZC/TO0 wird in unserem Versuch nicht verwendet. Durch das 1-Bit Nr. 7 im *Channel Control Word* ist aber der CTC so programmiert worden, daß er gleichzeitig mit dem 1-Signal am Ausgang ZC/TO0 an seinen INT-Ausgang ein Signal schaltet, das für die CPU eine Interrupt-Anforderung bedeutet.

Das letzte der Befehls-Bytes zur Initialisierung des CTC ist ein *Interrupt Control Word*. Es hat eine ähnliche Bedeutung wie das entsprechende Befehls-Byte bei der Programmierung der PIO. Dem CTC wird damit das LOB der ersten Ablage-Adresse für die Anfangs-Adresse einer Interrupt-Service-Routine übergeben.

Damit die CPU einer Interrupt-Anforderung vom CTC nachkommen kann, wird in der Initialisierungs-Routine der CPU das HOB der Ablage-Adresse übergeben und das IFF1-Flipflop gesetzt.

Nach der Programmierung des CTC wird die CPU in den HALT-Zustand versetzt, in dem sie geduldig und ansonsten untätig auf eine Interrupt-Anforderung wartet. Wenn die eintrifft, nämlich immer dann, wenn der Zähler im CTC von fünf Peripherie-Impulsen leergezählt worden ist, dann wird die Interrupt-Service-Routine angesprungen, in der eine Klingel programmiert ist.

Aufgabe S 47.1

a) Programmieren Sie den CTC so, daß die Klingel nach jeder dreizehnten Betätigung der Taste 0 des PIO-Ports B ertönt.

b) Was müssen Sie tun, wenn nicht die Betätigungen der Taste 0, sondern die der Taste 6 abgezählt werden sollen?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 8.

Sie werden die Bedeutungen der einzelnen Bits im *Channel Control Word* in einem der nächsten Abschnitte kennenlernen. Dazu muß jedoch der interne Aufbau des CTC genauer bekannt sein.

Bei den Beispielen für die Verwendung des CTC haben wir bereits darauf hingewiesen, daß der CTC in zwei unterschiedlichen Betriebsarten arbeiten kann: Als reiner Ereigniszähler im sogenannten *Counter Mode* (wie im vorangegangenen Versuch) und als Zeitgeber (Uhr) im *Timer Mode*. In beiden Fällen wird ein interner Zähler im CTC mit einem bestimmten Wert geladen und dann leergezählt.

Die Kommandos zum Dekrementieren des Zählers kommen in den beiden Betriebsarten von unterschiedlichen Quellen. Im *Counter Mode*

liefert die Peripherie die Impulse, die den Zähler dekrementieren. Im *Timer Mode* müssen die Dekrementier-Impulse mit größter Regelmäßigkeit kommen. Sie stehen in einem Mikroprozessor-System in Form des Systems-Taktes mit Quarzgenauigkeit zur Verfügung.

Es sei bereits jetzt darauf hingewiesen, daß die Einschaltung einer der beiden Betriebsarten über das Bit Nr. 6 des *Channel Control Words* erfolgt. Der Wert 1 dieses Bits bewirkt die Arbeitsweise des Kanals als Ereigniszähler; der Wert Ø läßt den Kanal als Zeitgeber arbeiten.

In beiden Betriebsarten ist für den internen Zähler des Kanals der entscheidende Augenblick jeweils dann gekommen, wenn sein Inhalt bis zum Wert Ø hinuntergezählt worden ist. Genau dann schaltet der Kanal ein kurzzeitiges 1-Signal an seinen Anschluß ZC/TO. Wenn der Kanal für Interrupts programmiert worden ist, dann wird außerdem eine Interrupt-Anforderung an die CPU geschickt.

Und was geschieht dann anschließend? Vielleicht haben Sie sich diese Frage bereits im Anschluß an den letzten Versuch gestellt, wenn Sie das Programm genauer angeschaut haben. Denn eigentlich sollte man die Notwendigkeit erwarten, den Zähler mit einem neuen *Time Constant Word* nachladen zu müssen, nachdem er einmal leergezählt worden ist.

Genau das ist aber nicht notwendig, und das trägt zur einfachen Programmierung des CTC wesentlich bei:

> Nach dem Leerzählen des internen Zählers eines Kanals wird der Zähler automatisch mit dem Wert nachgeladen, der bei der Programmierung des Kanals mit dem *Time Constant Word* übergeben worden ist. – Eine Änderung dieses Werts ist nur durch neue Programmierung mit einem *Channel Control Word* und einem *Time Constant Word* möglich.

Die Programmierung des Z 80 CTC

In den vorangegangenen Abschnitten haben Sie die Verwendbarkeit und den inneren Aufbau des CTC kennengelernt. Auch das Prinzip der Programmierung haben wir Ihnen vorgestellt. Bereits dabei hat sich gezeigt, daß die Programmierung ganz ähnlich der der Z 80 PIO ist, aber offensichtlich auch wesentlich einfacher.

Wenn der CTC im *Timer Mode* nichts anderes tun soll, als in programmierbaren, regelmäßigen Abständen Impulse an die Peripherie zu liefern (Bild S41.1), dann ist seine Programmierung ganz besonders einfach. An die Adresse des CTC-Kanals, der diese Aufgabe übernehmen soll, braucht nur ein *Channel Control Word* geliefert zu werden, in dem die gestellte Aufgabe sozusagen bitweise beschrieben wird. Anschließend wird ein *Time Constant Word* programmiert, mit dem die zeitlichen Impuls-Abstände bestimmt werden.

Zur Programmierung des *Channel Control Words* und des *Time Constant Words* werden jeweils vier Byte benötigt (vgl. die INIT-Routine auf der Seite L 45). Die ganze Programmierung kann also mit acht Byte erledigt werden.

Bei voller Ausnutzung der Fähigkeiten des CTC ist der Programmier-Aufwand etwas größer. Jeder CTC-Kanal ist sozusagen ein in sich abgeschlossenes Gebilde. Jeder der Kanäle muß deshalb auch getrennt mit einem *Channel Control Word* und einem *Time Constant Word* programmiert werden. Ein *Interrupt Vector Word* braucht jedoch nur einmal an den CTC geliefert zu werden. Es hat dann für alle vier Kanäle Gültigkeit.

In diesem Abschnitt wird die Codierung des *Channel Control Words*, des *Time Constant Words* und des *Interrupt Control Words* vorgestellt.

Das Channel Control Word

Das *Channel Control Word* spielt beim CTC eine ähnliche Rolle wie das *Interrupt Control Word* bei der PIO. Je nach Art des Bitmusters stellt es einen Ein- oder einen Zwei-Byte-Befehl dar, bei dem das evtl. folgende, zweite Byte als *Time Constant Word* bezeichnet wird. Die Anzahl der Bits, die Informationen für die Arbeitsweise des Peripherie-Bausteins enthalten, ist aber wesentlich größer als die des entsprechenden Befehls-Bytes bei der PIO: Bei der Zusammenstellung des *Channel Control Words* müssen die Werte von sieben Bits beachtet werden. Das Bit Nr. Ø enthält mit dem Wert 1 die Kennung, an welcher der CTC das *Channel Control Word* erkennt.

Im Bild S50.1 haben wir das Bild S45.1 noch einmal wiederholt. Es zeigt die Bedeutung der Bits in den drei möglichen Befehls-Bytes für den CTC. – Sehen Sie sich zunächst bitte das *Channel Control Word* an. Die Bedeutung der einzelnen Bits soll der Reihe nach vorgestellt werden.

Bit Nr. Ø: Befehlskennung

Der CTC kennt als Befehle eigentlich nur das *Channel Control Word* und das *Interrupt Vector Word*. Das *Time Constant Word* stellt keinen eigenständigen Befehl dar; wenn es programmiert wird, dann gehört es als zweites Byte zum *Channel Control Word* und wird als solches automatisch erkannt.

Die zwei möglichen Befehle für den CTC können durch den Wert eines einzigen Bits auseinandergehalten werden. Für dieses Bit Nr. Ø gelten folgende Bedeutungen:

Ø: Kennung für das *Interrupt Vector Word*
1: Kennung für das *Channel Control Word*

Bit Nr. 1: Reset

Beim CTC muß unterschieden werden zwischen einem Hardware-Reset, der über ein Ø-Signal am RESET-Anschluß des Bausteins ausgelöst wird, und einem Software-Reset, den ein *Channel Control Word* mit dem Wert 1 des Bits Nr. 1 verursacht.

Ein Hardware-Reset versetzt den CTC in den vollkommen unprogrammierten Zustand, in dem alle internen Register gelöscht sind. Außerdem wird die Verbindung des CTC mit dem Datenbus des Systems intern abgeschaltet. Nach einem solchen Reset muß der CTC vollkommen neu programmiert werden.

Ein Software-Reset bezieht sich immer nur auf den Kanal, dem das *Channel Control Word* mit einem Reset-Kommando übermittelt wird. In diesem Kanal wird das Leerzählen des Abwärts-Zählers durch den Reset unterbrochen. Die übrigen Eigenschaften des Kanals stellen sich entsprechend den Bit-Werten im *Channel Control Word* ein. Das Leerzählen des Abwärts-Zählers wird erst dann fortgesetzt, wenn eine neue Zeitkonstante in das *Time Constant Register* (Bild H 55.1) eingetragen worden ist.

Der Inhalt des Abwärts-Zählers bleibt bei einem Software-Reset unverändert. Nach der Übermittlung der neuen Zeitkonstante wird bei Wiederaufnahme des Lehrzählens zunächst der alte Inhalt des Abwärts-Zählers hinuntergezählt.

Ø: Kein Software-Reset
1: Software-Reset

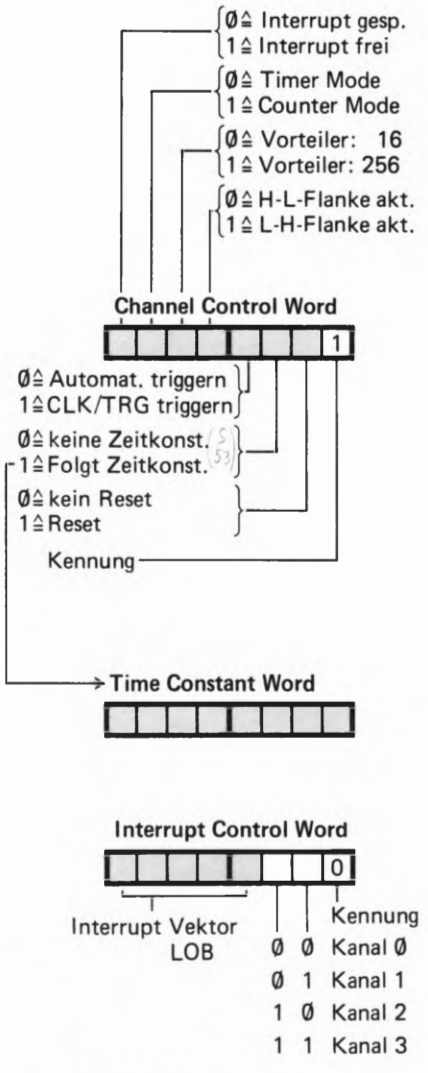


Bild S 50.1
Wiederholung des Bildes S 45.1. Mit diesen Befehls-Bytes wird der Z 80 CTC programmiert.

Bit Nr. 2: Folgt Zeitkonstante?

Es wurde bereits mehrfach darauf hingewiesen, daß das *Channel Control Word* ein Ein- oder ein Zwei-Byte-Befehl sein kann, je nachdem, ob ein *Time Constant Word* folgt oder nicht.

Die Arbeitsweise eines CTC-Kanals kann während des Betriebs jederzeit durch die Übermittlung eines neuen *Channel Control Words* geändert werden. Dem *Channel Control Word* braucht dann kein *Time Constant Word* zu folgen. Wenn die Zeitkonstante geändert werden soll, dann muß diese allerdings immer in Verbindung mit einem *Channel Control Word* programmiert werden.

Mit dem Wert des Bits Nr. 2 des *Channel Control Words* wird dem CTC mitgeteilt, ob das nächste übermittelte Byte ein *Time Constant Word* sein wird oder nicht, ob also eine Umprogrammierung des Kanals ohne Änderung der Zeitkonstante vorgenommen wird.

Ø: Es folgt kein *Time Constant Word*
1: Es folgt ein *Time Constant Word*

Bit Nr. 3: *Trigger Mode*

Dieses Bit hat nur dann Bedeutung, wenn der CTC im *Timer Mode* arbeitet. Im *Counter Mode* hat das Bit Nr. 3 *don't-care*-Eigenschaften: Sein Wert kann beliebig sein.

Was es mit dem Trigger auf sich hat, wurde bereits auf der Seite H54 erläutert. Nach der Programmierung eines *Time Constant Words* hängt es vom Wert des Bits Nr. 3 im *Channel Control Word* ab, ob das Leerzählen des Abwärtszählers im angesprochenen Kanal sofort beginnt, oder ob das Leerzählen mit einem Signal von der Peripherie am Anschluß CLK/TRG gestartet wird.

Wenn dem Kanal als Bit Nr. 3 des *Channel Control Words* der Wert Ø übermittelt wird, dann beginnt im angesprochenen Kanal sofort das Leerzählen. Beim Inhalt Ø des Zählers wird ein kurzzeitiges Signal an den Anschluß ZC/TO und bei entsprechender Programmierung eine Interrupt-Anforderung an die CPU geschickt. Gleichzeitig wird der Abwärts-Zähler aus dem *Time Constant Register* nachgeladen. Das Leerzählen beginnt dann sogleich von neuem. Am Anschluß ZC/TO erscheint eine ununterbrochene Folge von Impulsen, deren zeitlicher Abstand durch das *Time Constant Word* bestimmt ist. – Bei programmiertem Interrupt erhält die CPU in regelmäßiger Folge Interrupt-Anforderungen. Der CTC-Kanal arbeitet also als Uhr.

Wird dem Kanal dagegen als Bit Nr. 3 des *Channel Control Words* der Wert 1 übermittelt, dann beginnt das erstmalige Leerzählen nicht sofort nach der Übermittlung des *Time Constant Words*, sondern erst dann, wenn die Peripherie an den CTC-Anschluß CLK/TRG ein Trigger-Signal geliefert hat.

Sowohl nach der automatischen Triggerung als auch nach der Triggerung über den Anschluß CLK/TRG arbeitet der Kanal solange ununterbrochen, bis er durch einen Reset angehalten wird.

Ø: Automatisches Triggern nach dem *Time Constant Word*
1: Triggern über den Anschluß CLK/TRG

Bit Nr. 4: Trigger Flanke

Mit dem Wert des Bits Nr. 4 kann die aktive Flanke des Signals am Anschluß CLK/TRG gewählt werden. Dieses Signal bewirkt im *Timer Mode* beim Wert 1 des Bits Nr. 3 den Start des Leerzählens des Abwärts-Zählers; im *Counter Mode* wird der Inhalt des Abwärts-Zählers dekrementiert.

Ø: Signal an CLK/TRG wird beim 1-Ø-Übergang wirksam.
1: Signal an CLK/TRG wird beim Ø-1-Übergang wirksam.

S

52

Bit Nr. 5: Einstellung des Vorteilers

Die Wirkung des Vorteilers wurde auf den Seiten H 55 und H 56 erläutert.

Ø: Der Vorteiler teilt den System-Takt durch 16
1: Der Vorteiler teilt den System-Takt durch 256

Bit Nr. 6: *Timer/Counter-Mode*

Mit dem Bit Nr. 6 im *Channel Control Word* wird eingestellt, ob der CTC-Kanal im *Timer-* oder im *Counter Mode* arbeiten soll.

Ø: Der CTC-Kanal arbeitet im *Timer Mode*
1: Der CTC-Kanal arbeitet im *Counter Mode*

Bit Nr. 7: *Interrupt Enable*

Je nach dem Wert dieses Bits im *Channel Control Word* wird dann, wenn der Abwärts-Zähler bis zum Inhalt Ø leergezählt worden ist, außer dem Signal am Anschluß ZC/TO eine Interrupt-Anforderung an die CPU geschickt, oder eine solche Interrupt-Anforderung wird unterbunden.

Das Bit Nr. 7 im *Channel Control Word* hat also die gleiche Wirkung wie das Bit Nr. 7 im *Interrupt Control Word* der PIO (vgl. Seite S 28 und Bild S 28.1).

Ø: Keine Interrupt-Anforderungen
1: Interrupt-Anforderung gleichzeitig mit Signal an ZC/TO

Aufgabe S 52.1

Tragen Sie in das Bild S 52.1 die Bitmuster für *Channel Control Words* ein, mit denen der CTC wie folgt programmiert wird:

a) Kein Software-Reset. Der CTC soll als Uhr arbeiten, die in regelmäßiger Folge Impulse an den ZC/TO-Anschluß des CTC liefert. Die Impulse sollen keine Interrupt-Anforderung auslösen. Die Ausgabe der Impulse soll mit einem HIGH-LOW-Signalübergang am CTC-Anschluß CLK/TRG von der Peripherie gestartet werden.

Die Impuls-Frequenz, also die zeitlichen Abstände der Impulse, wird dem CTC im Anschluß an das *Channel Control Word* übermittelt. Der Vorteiler muß den System-Takt durch 256 teilen.

b) Dem CTC wird ein Software-Reset übermittelt. – Nach dem Leerzählen des Abwärts-Zählers soll der CTC positive Peripherie-Impulse zählen, die ihm an den Anschluß CLK/TRG geliefert werden.

Jeweils nach dem Eintreffen einer vorgegebenen Anzahl von Impulsen soll die CPU eine Interrupt-Routine anspringen.

Die Anzahl von Peripherie-Impulsen, nach denen ein Interrupt ausgelöst werden soll, wird dem CTC im Anschluß an das *Channel Control Word* übermittelt.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 9.

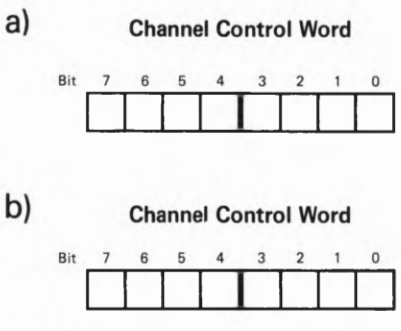


Bild S 52.1
Zu Aufgabe S 52.1: Tragen Sie bitte die Bitmuster der Channel Control Words entsprechend den Teilaufgaben a) und b) ein!

Das Time Constant Word

Das *Time Constant Word* stellt bei Bedarf das an den CTC zu übermittelnde zweite Byte des *Channel Control Words* dar. Es ist daher nicht notwendig, dieses Befehls-Byte mit einem eigenen Kennungs-Bit zu versehen. Sämtliche acht Bits dieses Bytes stehen zur Codierung der Zeitkonstanten zur Verfügung.

Sicher haben Sie sich schon gefragt, wann denn eigentlich die Übermittlung einer Zeitkonstanten nicht notwendig ist. Bei der erstmaligen Programmierung muß dem CTC ganz sicher eine Zeitkonstante geschickt werden. Im *Counter Mode* hat dieses Byte zwar nicht die eigentliche Bedeutung einer Zeitkonstanten; es besorgt jedoch die Voreinstellung des *Time Constant Registers* (Bild H 55; vgl. auch Seite S 56).

Auch nach einem Software-Reset muß dem *Channel Control Word* ein *Time Constant Word* folgen, damit der CTC wieder ordnungsgemäß arbeiten kann (vgl. Seite S 50).

Denkbar ist es aber, daß aus irgendeinem Grunde z. B. irgendwann innerhalb eines Programms der im *Counter Mode* betriebene CTC nicht mehr auf die LOW-HIGH-Flanke eines Peripherie-Signals reagieren soll, sondern auf die HIGH-LOW-Flanke. Das kann durch die Übermittlung eines neuen *Channel Control Words* geschehen, in dem der Wert des Bits Nr. 4 von dem des vorher programmierten *Channel Control Words* abweicht. Wenn in diesem neuen Befehls-Byte dann zusätzlich das Bit Nr. 2 den Wert Ø hat (vgl. das Bild S 50), dann bleibt der im *Time Constant Register* abgelegte Wert unverändert, und der CTC kann – bis auf die jetzt programmierte Änderung – wie vorher weiter arbeiten.

Die Überlegungen, die der Bemessung des *Time Constant Words* zugrunde liegen, haben Sie bereits ab der Seite H 55 kennengelernt. Bei der Programmierung des CTC ist das aber wenig hilfreich. Sie wollen ja normalerweise nicht wissen, welchen Abstand zwei ZC/TO-Impulse eines für den *Timer Mode* fertig programmierten CTCs haben. Viel interessanter ist die umgekehrte Rechnung: Mit welchem Wert muß das *Time Constant Register* über ein *Time Constant Word* geladen werden, wenn ein bestimmter Impuls-Abstand vorgegeben ist. Und: Ist dieser zeitliche Abstand überhaupt bei einfacher Programmierung des CTC zu erreichen?

Die Bedeutung des Vorteilers, der den System-Takt wahlweise durch 16 oder durch 256 teilt, wurde bereits ab der Seite H 55 erläutert. Hingewiesen werde aber noch einmal darauf, daß eine Berechnung des *Time Constant Words* im hier angesprochenen Sinne nur dann interessant ist, wenn der CTC im *Timer Mode* arbeitet, wenn der Abwärts-Zähler eines CTC-Kanals also mit Hilfe des System-Takts dekrementiert wird. Im *Counter Mode* ist der Vorteiler ganz uninteressant.

Es ist einleuchtend, daß die im *Timer Mode* des CTC erreichbaren Zeiten für das Leerzählen des Abwärts-Zählers stark von der Frequenz des System-Takts abhängig sind. Je höher diese Frequenz ist, um so schneller ist der Abwärts-Zähler jeweils leergezählt.

Wir gehen bei unseren Rechnungen von der Taktfrequenz 1,79 MHz im Micro-Professor aus. In diesem System folgen die Takt-Impulse in einem zeitlichen Abstand von etwa 0,56 µs. (Vgl. Seite H 55.)

Ohne den Rechengang anzugeben, stellen wir Ihnen zwei Gleichungen vor, mit denen Sie sehr einfach den (dezimal dargestellten!) Wert des *Time Constant Words* ermitteln können, wenn Sie eine bestimmte Zeit für das Leerzählen des Abwärtszählers vorgeben. Die erste der beiden Gleichungen gilt für den Fall, daß der Vorteiler die Taktfrequenz des Systems durch 16 teilt; die zweite gilt für eine Teilung durch 256.

Teilung durch 16

$$TCW = \frac{Tv}{ms} \quad 111{,}61$$

Teilung durch 256

$$TCW = \frac{Tv}{ms} \quad 6{,}975$$

In diesen beiden Gleichungen ist TCW die Abkürzung für das in Millisekunden angegebene **T***ime* **C***onstant* **W***ord*. Beachten Sie bitte, daß die Gleichungen die Zeitkonstante dezimal dargestellt liefern. Vor der Programmierung muß die Dezimal-Darstellung noch in die Sedezimal-Darstellung umgewandelt werden.

Der höchstmögliche Wert des *Time Constant Words* ist 255 = FFH, der kleinste Wert ist 1. – Wichtig ist zu wissen, daß der kleinste Wert nicht etwa das Byte ØØ ist! Das Byte ØØ wird vom CTC so interpretiert, als wäre der Wert (1)ØØH = 256 eingegeben worden.

Bei der Frequenz 1,79 MHz des System-Takts im Micro-Professor liegen die erreichbaren Zeiten für das Leerzählen des Abwärts-Zählers in folgenden Bereichen:

Kürzeste Zeit	Längste Zeit	Vorteiler
8,96 µs ..... 2,29 ms		1 : 16
Ø, 143 ms ..... 36,7 ms		1 : 256

Wenn Sie diese Aufstellung nicht beachten und das *Time Constant Word* für eine Zeit berechnen, die außerhalb der angegebenen Minimal- bzw. Maximal-Werte liegt, dann erhalten Sie einen Wert für das *Time Constant Word*, der kleiner als 1 oder größer als 256 ist. Das ist dann ein Zeichen dafür, daß diese Zeit bei der gewählten Einstellung des Vorteilers nicht erreicht werden kann. Wir werden Ihnen in einem späteren Abschnitt noch Beispiele dafür zeigen, wie man mit Hilfe des CTC auch längere Zeiten einstellen kann.

Aufgabe S 54.1

Schreiben Sie bitte eine Befehlsfolge in Form eines Unterprogramms zur Programmierung des CTC-Kanals 1 nach dem Muster der INIT-Routine auf der Seite L 45 an, die folgendes bewirkt:

Der Kanal erhält einen Software-Reset. – Die Peripherie startet mit einem HIGH-LOW-Übergang am CTC-Anschluß CLK/TRG1 die

Ausgabe einer Impuls-Wechselspannung mit einer Frequenz von 50 Hz am CTC-Anschluß ZC/TO1. – Es soll kein Interrupt ausgelöst werden.

Die Angabe von Kommentaren in der Befehls-Auflistung ist nicht notwendig. Es brauchen auch nur die mnemonischen Codes der Befehle und deren Operanden angeschrieben zu werden. – Das Programm „weiß" bereits, daß CTC1 die Port-Adresse des CTC-Kanals 1 ist.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 9.

Das Interrupt Control Word

Die in diesem Abschnitt bisher besprochenen Befehls-Bytes für den CTC müssen bei Bedarf für jeden der vier Kanäle des CTC getrennt programmiert werden. Anders ist es beim *Interrupt Control Word*, das auch bei der Programmierung mehrerer Kanäle nur ein einziges Mal an den CTC geschickt werden muß. (Vgl. Seite H 60.)

Im Bild S 56.1 finden Sie zwei Darstellungen von Vektorfeldern, in denen die Anfangsadressen von Interrupt-Service-Routinen für die Kanäle Ø bis 3 des CTC abgelegt sind. Wir interessieren uns zunächst nur für das Teilbild a.

Vergleichen Sie die Darstellung bitte mit dem Bild S 38.1, in dem das Vektorfeld der Anfangsadressen von Interrupt-Service-Routinen für die PIO-Ports A und B dargestellt ist. Wie in diesem Bild, so ist auch im Bild S 56.1a die Zusammensetzung der Adressen für die Ablage-Speicherzellen gezeigt: Das allen Adressen gemeinsame HOB steht im I-Register der CPU; das LOB liefert der CTC-Kanal, der den Interrupt anfordert.

Im Bild S 56.1a sind links vom Vektorfeld die Adressen sedezimal angeschrieben. Rechts ist die gleiche Adresse noch einmal als Bitmuster angegeben. Weil das Vektorfeld im Prinzip an einer beliebigen Stelle im Speicher angelegt werden kann, sind für das Adressen-HOB und für das höherwertige Halbbyte des LOB Werte X eingetragen.

Im Zusammenhang mit der Programmierung des CTC interessiert natürlich das Adressen-LOB besonders, denn dieses LOB muß dem CTC als *Interrupt Vector Word* mitgeteilt werden. Darin hat das ganz rechts stehende Bit Nr. Ø als Kennung für dieses Befehls-Byte immer den Wert Ø (vgl. das Bild S 50.1). Was das bedeutet, haben wir im Zusammenhang mit dem *Interrupt Vector Word* der PIO erläutert: Die erste der beiden Ablage-Adressen muß immer ohne Rest durch 2 teilbar sein.

Bis hierher besteht eigentlich keinerlei Unterschied zwischen einem der PIO und dem CTC bei der Programmierung zu übermittelnden *Interrupt Vector Word.* Ein Unterschied ist Ihnen allerdings schon bekannt: Bei der Programmierung der PIO muß jedem Port das *Interrupt Vector Word* getrennt übermittelt werden; beim CTC wird nur einmal ein *Interrupt Vector Word* programmiert, das dann für alle vier Kanäle Gültigkeit hat. Wie ist das möglich?

S
56

Wir wollen annehmen, die erste Adresse des Vektorfeldes wäre XXXØH; bei dieser Adresse ist das Anfangs-Adressen-LOB der Interrupt-Service-Routine abgelegt, die vom CTC-Kanal Ø aufgerufen wird. (Das HOB der Anfangs-Adresse ist bei XXX1H abgelegt. Den Inhalt dieser Adresse holt sich die CPU bei einer Interrupt-Anforderung automatisch.) Bei der Programmierung wird dem CTC das Byte XØH mit dem Bitmuster XXXX ØØØØ als *Interrupt Vector Word* für den Kanal Ø übermittelt. Am Wert Ø des im Bitmuster ganz rechts stehenden Bits Nr. Ø erkennt der CTC, daß es sich um ein *Interrupt Vector Word* handelt.

Jetzt kommt der Trick. Nachdem der CTC das *Interrupt Vector Word* als solches erkannt hat, verwirft er intern sofort die drei niederwertigen Bits Nr. Ø, Nr. 1 und Nr. 2. Bei einer dann folgenden Interrupt-Anforderung durch einen seiner drei Kanäle trägt er selbständig wieder für das Bit Nr. Ø den Wert Ø ein. Er weiß ja, daß die CPU an dieser Stelle immer den Wert Ø erwartet.

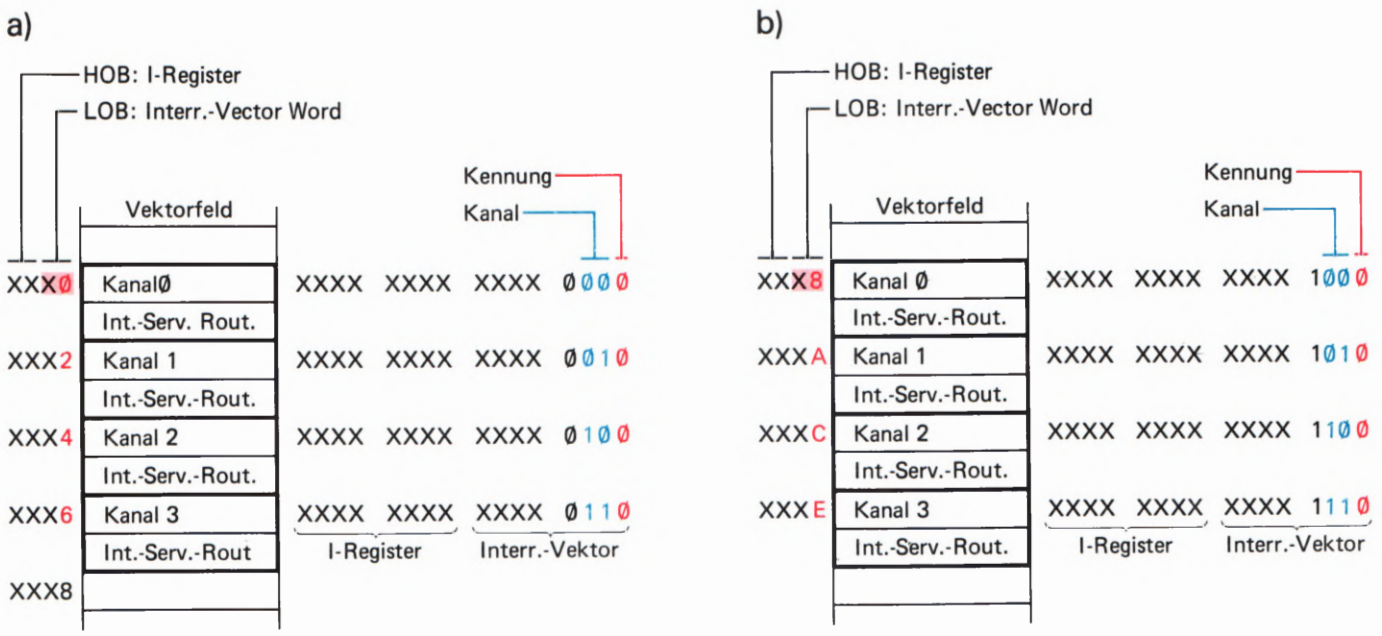
Für die Bits Nr. 1 und Nr. 2 setzt er dann ganz automatisch dual codiert die Nummer des Kanals ein, der den Interrupt anfordert. Im Bild S 56.1 haben wir diese beiden Bits blau markiert.

Noch einmal: Von den acht Bits des *Interrupt Vector Words* registriert der CTC nur die oberen fünf Bits Nr. 3 bis Nr. 7. Das Bit Nr. Ø braucht er, um das *Interrupt Vector Word* als solches zu erkennen; die Bits Nr. 1 und Nr. 2 interessieren ihn überhaupt nicht. Er fügt sie zusammen mit dem Wert Ø des Bits Nr. Ø bei einer Interrupt-Anforderung selbständig an die registrierten Bits Nr. 3 bis Nr. 7. Dieses so vervollständigte Byte teilt er der CPU als Interrupt-Vektor mit.

Es zeigt sich, daß bei der Programmierung des *Interrupt Vector Words* die Bits Nr. 1 und Nr. 2, die im Bild S 56.1 blau markiert sind, *don't care*-Charakter haben. Das (rot markierte) Bit Nr. Ø hat ausschließlich den Charakter eines Kennungs-Bits.

Bild S 56.1
Das Interrupt Vector Word des CTC kann eine der Formen XØ und X8 haben.

Entscheidend sind bei der Programmierung des *Interrupt Vector Words* die Bits Nr. 3 bis Nr. 7. Nur mit diesen Bits kann der Programmierer



dem CTC die Adresse der ersten Ablage-Speicherzelle im Vektorfeld des CTC mitteilen. Die restlichen Adressen der Ablage-Speicherzellen für die Interrupt-Service-Routine generiert dann der CTC selbständig. Es ist also nicht notwendig, dem CTC mehr als ein einziges *Interrupt Vector Word* mitzuteilen.

Unsere Überlegungen führen zu einem interessanten Schluß. Für die erste CTC-Vektorfeld-Adresse ist das HOB und das höherwertige Halbbyte des LOB frei wählbar (Werte X im Bild S 56.1). Im niederwertigen Halbbyte des Adressen-LOBs ist nur das Bit Nr. 3 frei wählbar; es kann den Wert Ø oder den Wert 1 haben. Über die Bits Nr. Ø bis Nr. 2 kann nicht verfügt werden.

Sehen Sie sich jetzt bitte die beiden Teilbilder S 56.1a und b im Zusammenhang an! Sie erkennen:

> Das Vektorfeld für die Ablage der Anfangs-Adressen der CTC-Interrupt-Service-Routinen muß entweder bei einer Adresse XXXØH (Bild S 56.1a) oder bei einer Adresse XXX8H (Teilbild b) beginnen.

Man kann versuchen, dem CTC als *Interrupt Vector Word* das Byte 31H mit dem Bitmuster ØØ11 ØØØ1 zu übermitteln. Dieses Byte würde der CTC wegen des Werts 1 des Kennungs-Bits Nr. Ø als *Channel Control Word* interpretieren (vgl. das Bild S 50.1), aber niemals als *Interrupt Vector Word.*

Wenn man versucht, das CTC-Vektorfeld bei der Adresse 1834H (Bitmuster ØØØ1 1ØØØ ØØ11 Ø1ØØ) beginnen zu lassen, dann erkennt der CTC zwar am Wert Ø des Kennungs-Bits Nr. Ø ein *Interrupt Vector Word*, wenn ihm das Adressen-LOB 32H übermittelt wird. Er registriert aber von diesem Byte nur die im Bitmuster stärker gedruckten Bits. Bei einer Interrupt-Anforderung z. B. des CTC-Kanals 1 schickt er dann an die CPU den Interrupt-Vektor 32H (Bitmuster ØØ11 ØØ1Ø). Weil ab der Adresse 1832H jedoch keine ordentliche Anfangsadresse für eine Interrupt-Service-Routine abgelegt ist, läuft das Programm bei einer solchen Interrupt-Anforderung mit einiger Sicherheit „in die Wüste".

Aufgabe S 57.1

Ein Programm, in dem der CTC für Interrupt-Anforderungen programmiert ist, schließt mit dem Befehl JP START ab. Es ist in diesem Zusammenhang ganz unwichtig, welche Befehle das Programm zwischen dem Label START und JP-Befehl enthält. Wesentlich ist die Feststellung, daß der Operationscode des JP-Befehls bei der Adresse 186E steht.

a) Wie heißt das *Interrupt Vector Word* für den CTC, wenn möglichst wenige Adressen unbenutzt bleiben sollen?

b) Kann ein anderes *Interrupt Vector Word* verwendet werden, wenn im CTC der Kanal Ø nicht verwendet wird?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 10.

Ein einfaches Programm mit CTC-Interrupt

Versuch S58.1

Interrupt-gesteuertes Lauflicht

Schließen Sie die Peripherie-Leiterplatte an Ihr System an, und tasten Sie das ab der Seite L47 aufgelistete Interrupt-Lauflicht-Programm ein! Die Stifte für die steckbaren Brücken auf der Peripherie-Leiterplatte werden für diesen Versuch nicht bestückt.

Starten Sie das Programm bei der Adresse 18ØØH. Die Leuchtdioden an den Ausgängen des PIO-Ports B bilden ein Lauflicht von rechts nach links. Das ist nichts besonderes. Sehen Sie sich aber bitte das zugehörige Programm an!

Nach dem einmaligen Aufruf der Initialisierungs-Routine INIT wird das Hauptprogramm bearbeitet, das aus einer Schleife mit nur zwei Befehlen besteht. Die CPU wartet zuerst im HALT-Zustand auf einen Interrupt. Nach dem Abarbeiten der Interrupt-Service-Routine springt das Programm wieder zum HALT-Befehl. Die CPU wird also die meiste Zeit im HALT-Zustand verbringen.

In der INIT-Routine wird zuerst der PIO-Port B für die Betriebsart 3 programmiert, und dann werden alle Anschlüsse des Ports B mit dem *I/O Register Control Word* zu Ausgängen gemacht. – Interrupts braucht die PIO nicht zu bearbeiten.

Anschließend folgt die Programmierung des CTC-Kanals Ø. Er arbeitet im *Timer Mode* und fordert jeweils nach Ablauf der über das *Time Constant Word* eingestellten Zeitkonstante einen Interrupt an. Der Vorteiler wird im Interesse nicht zu schnell aufeinanderfolgender Interrupts auf eine Teilung des Systemtakts durch 256 eingestellt. – Mit dem *Interrupt Vector Word* wird dem CTC das LOB der ersten Adresse im Vektorfeld übergeben. In diesem LOB mit dem Bitmuster ØØ11 1ØØØ (Byte 38H) hat – entsprechend unseren Überlegungen im vorhergehenden Abschnitt – das Bit Nr. 3 den Wert 1.

Die Rechnung auf der Seite H56 hat gezeigt, daß beim Wert FF des *Time Constant Words* und bei einer Teilung des System-Takts durch 256 jeweils nach Ablauf von 37 ms bei der CPU ein Interrupt angefordert wird. Würde in der dann aufgerufenen Interrupt-Service-Routine das Lauflicht jeweils um eine Leuchtdiode weitergeschaltet, dann liefe der Lichtpunkt viel zu schnell.

Wir bedienen uns eines Tricks, um das Lauflicht langsamer zu machen: Nicht bei jeder Bearbeitung der Interrupt-Service-Routine wird der Lichtpunkt um eins weitergeschaltet, sondern erst bei jeder dritten. Dazu wird in der Interrupt-Service-Routine ein anfangs auf 3 eingestellter Zähler im C-Register dekrementiert. Solange dieser Zähler noch nicht den Wert Ø hat, wird die Routine ohne weitere Aktion abgeschlossen. Wenn der Zähler aber bei jedem dritten Interrupt auf null angelangt ist, wird er im SCHIEB-Unterprogramm wieder auf den Wert 3 gesetzt. Dann wird der Wert 1 im Leuchtdioden-Bitmuster um eine Stelle nach links geschoben und dieses neue Bitmuster über den PIO-Port B ausgegeben.

PIO- und CTC-Interrupts

Die Beschäftigung mit der Z 80 PIO und dem Z 80 CTC hat gezeigt, daß diese Bausteine ihr wirkliches Können erst im Zusammenhang mit der Interrupt-Programmierung zeigen. Das Prinzip des Interrupts haben Sie bei verschiedenen Versuchen kennengelernt. Wir haben Ihnen aber noch nicht gezeigt, daß bei der Interrupt-Programmierung von der CPU tatsächlich zwei verschiedene Aufgaben sozusagen gleichzeitig und unabhängig voneinander erledigt werden können.

Abgesehen von den Programmen zur Drucker-Steuerung ist es nicht unser Anliegen, Ihnen technisch verwertbare, fertige Programme zu liefern. Unsere Versuchs-Programme sollen Ihnen vielmehr Anregungen bieten, entsprechende Programme selbst zu konzipieren.

In diesem Abschnitt wollen wir Ihnen zeigen, daß mit der Interrupt-Programmierung gleichzeitig sehr unterschiedliche Aufgaben gelöst werden können. In Versuchs-Programmen zeigen wir Ihnen die Kombination von PIO- und CTC-Interrupts.

Eine schaltbare Sekunden-Uhr

In den bisher vorgestellten Versuchs-Programmen mit Interrupt hat die CPU im HALT-Zustand – sozusagen im Urlaub – darauf gewartet, daß sie per Interrupt kurzzeitig zu einer bestimmter Tätigkeit herangezogen wurde. Nach Erledigung dieser Tätigkeit hat sie jeweils ihren Urlaub fortgesetzt.

Hier zeigen wir Ihnen ein Programm, in dem die CPU mit der Generierung des bereits mehrfach verwendeten Lauflichts beschäftigt wird. Diese Tätigkeit soll sie unterbrechen, wenn eine Interrupt-Anforderung vorliegt. Nach Erledigung der Interrupt-Service-Routine soll sie dann ihre normale Tätigkeit wieder aufnehmen. Dem Programm liegt folgender Gedankengang zugrunde:

Das Lauflicht wird ohne Interrupt programmiert. Über den PIO-Port B wird ein Bitmuster mit einem 1-Bit aus dem Akkumulator ausgegeben. Das Programm durchläuft dann eine Warte-Schleife, schiebt anschließend das 1-Bit im Akkumulator um eine Stelle nach links und gibt das Bitmuster erneut über den PIO-Port B aus.

Dieses Hauptprogramm ist wahrlich primitiv. Es kann aber von zwei unterschiedlichen Interrupt-Anforderungen unterbrochen werden.

Den einen dieser Interrupts verursacht die Taste Nr. Ø am PIO-Port A. Mit der Betätigung dieser Taste kann eine Funktion des CTCs zunächst eingeschaltet und bei der nächstfolgenden Betätigung wieder ausgeschaltet werden. Die von der Taste aufgerufene Interrupt-Service-Routine fragt ein Flag ab, dessen Zustand sie davon unterrichtet, ob die Funktion des CTC ein- oder ausgeschaltet ist. (Das Flag ist in einer Speicherzelle abgelegt.)

Wenn die Interrupt-Service-Routine des PIO-Ports A feststellt, daß die Funktion des CTC ausgeschaltet ist (Flag = 1), dann werden Interrupt-Anforderungen durch den CTC-Kanal Ø freigegeben. Vorher

wird aber noch das Flag gelöscht, damit bei der nächsten Betätigung der Taste in der Interrupt-Service-Routine die Interrupt-Anforderungen durch den CTC-Kanal Ø wieder gesperrt werden können.

Der Vorgang läuft genau mit der entgegengesetzten Wirkung ab, wenn die Interrupt-Service-Routine des PIO-Ports A feststellt, daß die Funktion des CTC eingeschaltet ist. Jetzt wird zuerst das Flag gesetzt, damit bei der nächsten Tasten-Betätigung die CTC-Funktion von neuem eingeschaltet werden kann. Anschließend werden Interrupt-Anforderungen des CTC gesperrt und damit die Funktion des CTC ausgeschaltet.

Die zweite Interrupt-Service-Routine ruft der Kanal Ø des CTC auf, wenn dieser für Interrupt-Anforderungen freigegeben worden ist. Der Kanal wird in der Initialisierungs-Routine zum Arbeiten im *Timer Mode* programmiert. Das *Time Constant Word* ist so bemessen, daß nach jeweils 218 Interrupts eine Sekunde vergangen ist. Der Vorteiler des Kanals wird dazu auf eine Teilung des Systems-Takts durch 256 eingestellt.

In der Interrupt-Service-Routine des CTC-Kanals Ø wird zunächst der Inhalt eines im Speicher angelegten Zählers dekrementiert. Dieser Zähler wird bei der Programm-Eingabe mit dem Wert 218 (= DAH) geladen. Wenn der Zähler noch nicht auf den Wert ØØ hinunter gezählt worden ist, wird die Interrupt-Service-Routine ohne weitere Aktionen wieder verlassen. Ist der Inhalt des Zählers jedoch beim Wert ØØ angekommen, dann ist das ein Zeichen dafür, daß 218 CTC-Interrupts abgelaufen sind, daß also gerade eine Sekunde vorbei ist. In diesem Fall wird der Zähler für den Ablauf der nächsten Sekunde erneut mit dem Wert 218 geladen. Dann wird mit Hilfe einer Routine des Micro-Professor-Betriebsprogramms ein kurzes akustisches Signal ausgegeben. Die Interrupt-Service-Routine des CTC ist damit abgeschlossen.

Versuch S 60.1

Lauflicht mit schaltbarer Sekunden-Uhr

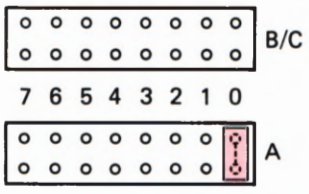


Bild S 60.1
Bestücken Sie die mit Ø bezeichneten Stifte des Ports A für den Versuch S 60.1 mit einer steckbaren Brücke.

Schließen Sie die Peripherie-Leiterplatte an Ihr System an, und tasten Sie das auf den Seiten L 51 bis L 56 aufgelistete Programm ein. – Setzen Sie eine steckbare Brücke auf die mit Ø bezeichneten Stifte für den Port A (Bild S 60.1) und starten Sie das Programm bei der Adresse 18ØØH.

Zunächst geschieht nichts anderes, als daß das Lauflicht über die zum PIO-Port B gehörenden Leuchtdioden startet. – Betätigen Sie kurzzeitig die zum PIO-Port A gehörende Taste Nr. Ø! Jetzt dauert es genau eine Sekunde, und ab dann hören Sie in Sekunden-Abständen kurze Piep-Signale aus dem Lautsprecher Ihres Systems. – Mit einer neuerlichen Betätigung der Taste Nr. Ø können Sie die Sekunden-Meldung wieder abschalten und durch eine weitere Betätigung dieser Taste auch wieder einschalten.

Die vom Programm bestimmte Funktion der Taste Nr. Ø wird in der Fach-Literatur als *Toggle* (sprich: toggel) bezeichnet. Es wird damit das feste Einstellen (und wieder Lösen) eines bestimmten Zustands bezeichnet.

Beobachten Sie bitte das Verhalten des Lauflichts bei eingeschalteter Sekunden-Meldung! Sie werden keine Beeinflussung des Ablaufs feststellen. Das Programm ist also tatsächlich so konzipiert, daß zwei ganz verschiedene Programme unabhängig voneinander bearbeitet werden.

Daß die beiden Programme nichts miteinander zu tun haben, erkennen Sie auch ohne Kenntnis des Programmablaufs daran, daß kein deutlich erkennbarer Rhythmus zwischen dem Lauflicht und der Sekunden-Meldung besteht.

Das Lauflicht wird im Hauptprogramm gesteuert; die Sekunden-Meldung wird per Interrupt erledigt. Die dazu jedesmal aufgerufene Interrupt-Service-Routine ist so kurz, daß der Ablauf des Hauptprogramms praktisch nicht gestört wird.

Daß die beiden Funktionen letztlich von der gleichen CPU gesteuert werden, erkennen Sie sofort, wenn Sie den Ablauf der Sekunden-Routine künstlich verlängern. Das geschieht am einfachsten, wenn Sie das akustische Signal für die Sekunden-Meldung verlängern.

Halten Sie das Programm an und ändern Sie das Byte 28H bei der Adresse 1892H versuchsweise auf einen höheren Wert! Nach dem Start des Programms und dem Einschalten der Sekunden-Meldung stellen Sie dann fest, daß der Ablauf des Lauflichts bei jeder Sekunden-Meldung sichtlich verzögert wird.

Einzelheiten des Programms brauchen wir hier nicht zu erläutern. Wenn Sie das interessiert, dann geben Ihnen die Kommentare in der Programm-Auflistung Auskunft. – Wir haben im Programm einige Z 80 spezifische Befehle verwendet, durch welche das Programm übersichtlicher gestaltet werden konnte.

Genau ansehen sollten Sie sich die INIT-Routine mit der Programmierung der PIO und des CTC. Der PIO-Port A wird für die Weitergabe von Interrupt-Anforderungen von der Taste Nr. Ø eingerichtet. Der PIO-Port B hat einfach die Aufgabe eines Ausgabe-Ports.

Beim CTC-Kanal Ø werden Interrupt-Anforderungen bei der Programmierung gesperrt. Sie werden erst in der Interrupt-Service-Routine der PIO mit einem *Channel Control Word* ohne nachfolgende Zeitkonstante freigegeben.

Sehen Sie sich bitte auch die Anlage des Vektorfeldes mit den Anfangs-Adressen der Interrupt-Service-Routinen an! Der Interrupt-Vektor des CTC-Kanals Ø zeigt auf die Adresse 18CØ. Das letzte Programm-Byte steht bei der Adresse 18B6. Die Speicherzelle mit der Adresse 18B7 wird für das Flag in einer Programm-Erweiterung reserviert. Die Anfangs-Adresse der Interrupt-Service-Routine für den CTC hätte also auch bei der Adresse 18B8 abgelegt werden können.

Weil in unserem Programm mit Sicherheit keine Interrupt-Anforderungen von anderen CTC-Kanälen kommen, könnte der Interrupt-Vektor der PIO auf die Adresse 18BA zeigen.

Wir sind bei der Programmierung den sicheren Weg gegangen. Weil dann unterhalb der Ablage für die Anfangs-Adresse der CTC-Interrupt-Service-Routine noch Platz ist, lassen wir den Interrupt-Vektor der PIO auf die Adresse 18BE zeigen. Jetzt ist bei Bedarf für die Ablage

der Anfangs-Adressen weiterer CTC-Interrupt-Service-Routinen hinreichend Platz.

Sie werden feststellen, daß wir innerhalb des Programms einige Bereiche für Programm-Ergänzungen freigelassen haben. Das ist geschehen, damit Sie beim folgenden Versuch nicht das ganze Programm neu eingeben müssen.

S

62

Interrupt-Lauflicht und Sekunden-Uhr

Im Programm des vorangegangenen Abschnitts haben wir Ihnen die Verarbeitung von Interrupts gezeigt, die von unterschiedlichen Peripherie-Bausteinen angefordert wurden. Das zugehörige Programm ist eigentlich nicht sehr zufriedenstellend.

Der CTC ist besonders elegant einzusetzen, wenn es um die Vermeidung von Verzögerungs-Schleifen geht, in denen die CPU ansonsten nutzlos nichts anderes tut, als einen Zähler leer oder voll zu zählen. Und genau das geschieht im Unterprogramm WAIT des Lauflicht-Programms (vgl. Seite L 51).

Im Interrupt-Lauflicht-Programm (Seiten S 58 und L 47) haben Sie gesehen, wie ein Lauflicht ohne Verzögerungs-Schleife von CTC-Signalen gesteuert werden kann. Wenn wir dieses Prinzip in das Lauflicht-Programm mit der Sekunden-Uhr übernehmen, dann laufen im Programm alle Aktivitäten Interrupt-gesteuert. Der Effekt ist natürlich, daß die CPU dann wieder nicht weiß, was sie in ihrer „Freizeit" tun soll. Wir wollen uns nicht die Mühe machen, ein Freizeitgestaltungs-Programm aufzuziehen und schicken die CPU zwischenzeitlich einfach in den HALT-Zustand. Uns geht es ja nur um das Prinzip der Sache.

Im Programm ab der Seite L 51 haben wir bereits den Platz reserviert, den die zusätzlichen Routinen benötigen. Es handelt sich dabei um die Programmierung eines weiteren CTC-Kanals und um die Interrupt-Service-Routine für diesen Kanal, in der die Bedienung des Lauflichts vorgenommen wird.

Die Programmierung des CTC-Kanals (es wird der Kanal 1 verwendet) braucht nicht erläutert zu werden. Sie entspricht genau der Programmierung auf der Seite L 48. Mit einer Ausnahme: Das *Interrupt Vector Word* braucht nicht noch einmal ausgegeben zu werden. Das ist bereits – gültig für alle CTC-Kanäle – bei der Programmierung des Kanals Ø geschehen.

Auch die Interrupt-Service-Routine arbeitet genau so, wie Sie es beim Interrupt-gesteuerten Lauflicht bereits kennengelernt haben. Damit die Umschaltungen der Leuchtdioden nicht zu schnell hintereinander erfolgen, löst nicht jeder Kanal-1-Interrupt eine solche Umschaltung aus. Im Speicher wird ein Zähler für drei Interrupts angelegt, der in der Interrupt-Service-Routine dekrementiert wird. Eine Umschaltung der Leuchtdioden erfolgt immer erst dann, wenn dieser Zähler leergezählt worden ist. Der Zähler wird in der Initialisierungs-Routine erstmalig mit dem Byte Ø3 geladen.

Bei der ergänzenden Programmierung darf selbstverständlich nicht vergessen werden, in das Vektorfeld die Anfangs-Adresse der Interrupt-Service-Routine für den CTC-Kanal 1 einzutragen.

Weil die Bedienung des Lauflichts jetzt in einer Interrupt-Service-Routine und nicht mehr im Hauptprogramm vorgenommen wird, schrumpft das Hauptprogramm auf einen HALT-Befehl zusammen, der in einer Schleife immer wieder angesprungen wird.

Versuch S 63.1

Interrupt-Lauflicht mit schaltbarer Sekunden-Uhr

Für diesen Versuch muß Ihr System mit dem Programm für den Versuch S 60.1 geladen sein (Programm ab der Seite L 51).

Tasten Sie in das System die Bytes ein, die zu den ab der Seite L 57 aufgelisteten Ergänzungen und Änderungen gehören. – Achten Sie bitte sorgfältig darauf, daß die Bytes bei den richtigen Adressen eingetragen werden! Vor jedem Teil der Ergänzungen und Änderungen steht in der Liste eine ORG-Anweisung, der Sie entnehmen können, wo die Ergänzung oder Änderung beginnt. Die ursprünglichen Bytes des Hauptprogramms und der WAIT-Routine brauchen nicht gelöscht zu werden. Sie müssen nur ab der Adresse 18Ø3H die Bytes für die Befehle HALT und JR BLINK eintasten.

Starten Sie das Programm und probieren Sie es aus! Es muß genau die gleichen Funktionen haben, die Sie bereits vom Versuch S 60.1 her kennen. Wenn diese Funktionen ausgeführt werden, dann arbeitet das Programm richtig.

Überzeugen Sie sich in der Auflistung der Befehle davon, was Sie jetzt programmiert haben.

Serielle Daten-Ausgabe mit CTC-Interrupt

Auf der Seite H 8 haben wir angedeutet, daß man mit Hilfe der in diesem Lehrgang vorgestellten Peripherie-Bausteine auch elegant eine Schnittstelle für einen Bit-seriellen Datenverkehr programmieren kann. Wir wollen Ihnen hier zum Abschluß der Beschäftigung mit der Z 80 PIO und dem Z 80 CTC ein Programm vorstellen, daß die Bit-serielle Daten-Ausgabe in ähnlicher Weise besorgt wie das Programm ab der Seite L 3.

Lesen Sie bitte noch einmal ab der Seite H 4 über das Prinzip der seriellen Datenübertragung nach! Bei dem in diesem Zusammenhang vorgestellten Programm haben wir festgestellt, daß auf jeden Fall ein Modul WARTE verwendet werden muß. Dieser Modul sorgt dafür, daß das ausgegebene Signal hinreichend lange am Ausgang verfügbar ist. Gleichzeitig haben wir aber bereits schon dort darauf hingewiesen, daß der Mikroprozessor in einer solchen WARTE-Schleife ganz nutzlos seine Zeit vertut.

Die Verwendung der in unserem Lehrgang vorgestellten Peripherie-Bausteine bietet die Möglichkeit, eine serielle Schnittstelle auch ohne

Zeitverschwendung über einen an sich parallel arbeitenden Daten-Port zu programmieren. Wir benutzen dazu eine Interrupt-Programmierung.

Das ab der Seite L59 aufgelistete Programm ist in der angegebenen Form ein Demonstrations-Programm, also nicht direkt zur praktischen Verwendung geeignet. Die seriell auszugebenden Bits erscheinen als Bit Nr. 3 des PIO-Ports B und werden von der zugehörigen Leuchtdiode angezeigt. Damit diese Anzeige verfolgt werden kann, erfolgt die Ausgabe recht langsam: Die Bits folgen in einem zeitlichen Abstand von etwa einer halben Sekunde. Um die Sache noch deutlicher zu machen, zeigt das Programm auch jeweils das Bitmuster aller acht Daten-Bits eines ausgesendeten Bytes. Die einzelnen Bits dieses Bytes werden von links nach rechts in die Anzeige des PIO-Ports A hineingeschoben. Das Start-Ø-Bit und die beiden generierten Stop-1-Bits (vgl. das Bild H5.1) werden nicht gesondert angezeigt.

Der Ausgabe-Rhythmus unseres Programms wird vom CTC gesteuert. Er liefert im Abstand der Bit-Zeiten, also etwa jede halbe Sekunde einmal, eine Interrupt-Anforderung an die CPU. Die zugehörige Interrupt-Service-Routine ist allein für die Bereitstellung und serielle Ausgabe eines Bytes zuständig. Die CPU kann also zwischenzeitlich ein ganz anderes Programm bearbeiten.

Die langsame Ausgabe ist hier interessant, weil ein einzelner CTC-Kanal bekanntlich nicht in der Lage ist, einzelne Impulse in dem gewünschten, großen zeitlichen Abstand zu generieren. Auf der Seite S54 können Sie nachlesen, daß der größte zeitliche Abstand zwischen zwei CTC-Impulsen beim Micro-Professor 36,7 ms beträgt.

Bei der Programmierung der Sekunden-Uhr im Interrupt-Lauflicht-Programm wurde dem CTC-Kanal ein Software-Zähler nachgeschaltet, um auf größere Impuls-Abstände zu kommen. Im hier beschriebenen Programm wird es anders gemacht: Die beiden CTC-Kanäle 1 und Ø werden per Hardware in Reihe geschaltet.

Das Bild S65.1 zeigt das Prinzip. Der CTC-Kanal 1 wird im *Timer Mode* betrieben. Er wird von dem im Verteiler durch 256 geteilten System-Takt getriggert. Im Interesse einer möglichst langsamen Impuls-Frequenz wird er mit dem *Time Constant Word* ØØ geladen. (Vgl. Seite S54.) Seine Interrupt-Ausgabe wird gesperrt; die Ausgangs-Impulse stehen an seinem Anschluß ZC/TO1 zur Verfügung. Dieser Anschluß ist an die Klemmleiste der Peripherie-Leiterplatte geführt (Bild H9.1).

Der CTC-Kanal Ø wird für den *Counter Mode* und für die Ausgabe von Interrupt-Anforderungen programmiert. Die Trigger-Signale erhält er über seinen Anschluß CLK/TRG0, der ebenfalls an der Klemmleiste zugänglich ist. Die Programmierung mit dem *Time Constant Word* ØEH (= 14) sorgt dafür, daß eine Interrupt-Anforderung nach jedem 14. Trigger-Signal erfolgt. Das Bild S65.1 zeigt, wie der Kanal Ø durch die ZC/TO-Signale des Kanals 1 getriggert wird.

Sehen Sie sich bitte die Programmierung der beiden CTC-Kanäle auf der Seite L6Ø an! Sie werden die Befehlsfolge jetzt ohne weiteres verstehen. Beachten Sie bitte, daß dem CTC nur ein einziges *Interrupt Vector Word* übermittelt wird!

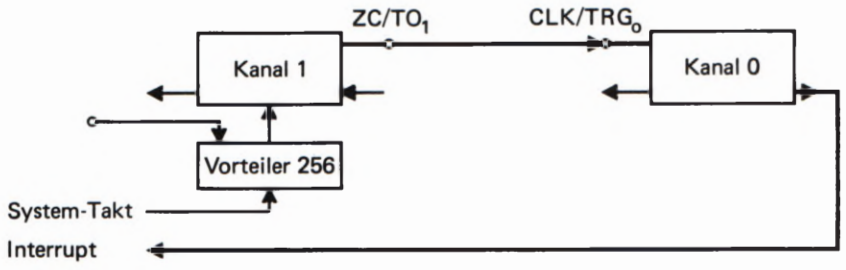


Bild S65.1
Die im CTC wirksame Zeitkonstante läßt sich durch die Reihenschaltung von zwei Kanälen scheinbar verlängern.

Bei dieser Gelegenheit können Sie auch gleich die Programmierung der beiden PIO-Ports auf der Seite L 60 ansehen. Über den Port A wird das Bitmuster des seriell auszusendenden Bytes angezeigt. Sämtliche Anschlüsse werden dazu als Ausgänge programmiert. – Die seriell auszugebenden Bits erscheinen am Anschluß Nr. 3 des Ports B. Der Anschluß Nr. 7 dient der Eingabe des CTS-Signals (vgl. Seite H 4). Dieser Anschluß wird deshalb für Eingaben programmiert; alle übrigen Anschlüsse können Ausgabe-Anschlüsse sein. – Für Interrupts braucht die PIO nicht programmiert zu werden.

Das Hauptprogramm für unseren Versuch ist wieder denkbar einfach: Statt die CPU mit einer bestimmten Aufgabe dauernd zu beschäftigen, lassen wir sie im HALT-Zustand auf CTC-Interrupts warten.

Die Interrupt-Service-Routine, die für die serielle Ausgabe allein zuständig ist, brauchen wir hier nicht zu erläutern. Ihre Funktion geht aus den Kommentaren in der Programm-Auflistung hervor.

Versuch S 65.1

Interrupt-gesteuerte serielle Ausgabe

Schließen Sie die Peripherie-Leiterplatte an Ihr System an. Verbinden Sie die Anschlüsse ZC/TO1 und CLK/TRG0 an der Klemmleiste durch eine kurze Leitung. – Setzen Sie eine steckbare Brücke auf die mit 7 bezeichneten Stifte für den Port B (Bild S 65.2).

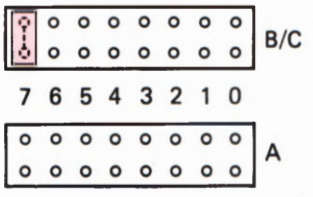


Bild S65.2
Für den Versuch S 65.1 müssen die zum PIO-Port B gehörenden Stifte mit der Bezeichnung 7 mit einer steckbaren Brücke bestückt werden.

Tasten Sie das ab der Seite L 59 aufgelistete Programm in Ihr System ein, und starten Sie es bei der Adresse 1800H! Die Leuchtdiode Nr. 3 des PIO-Ports B muß leuchten. Sie zeigt den Ruhe-Zustand des Ausgangs für die seriellen Daten an.

Geben Sie ein CTS-Signal durch die Betätigung der Taste Nr. 7 des PIO-Ports B ein! Die Leuchtdiode am Ausgang für die seriellen Daten wird verlöschen. Damit wird ein Ø-Start-Bit gemeldet. Die weitere Ausgabe von Daten-Bits können Sie an den Leuchtdioden des Ports A erkennen.

Wenn Sie die CTS-Taste nach dem Erscheinen des Start-Ø-Bits wieder losgelassen haben, dann wird nach der seriellen Ausgabe der acht Bits des ersten Daten-Bytes und der schließenden (nicht eigens angezeigten) zwei Stop-1-Bits kein weiteres Byte ausgegeben. Die acht Daten-Bits der ersten Bytes bleiben so lange in der Anzeige über den Port A stehen, bis Sie ein weiteres CTS-Signal eintasten.

Wenn Sie die CTS-Taste festhalten, dann erfolgt eine ununterbrochene Ausgabe aufeinanderfolgender Daten-Bytes mit jeweils einem Start-Ø-Bit und zwei Stop-1-Bits. – Das Programm gibt seine eigenen Befehls-Bytes ab der Adresse 1800H aus.

Das hier beschriebene Programm läßt sich selbstverständlich auch praktisch für die serielle Ausgabe von Bytes verwenden. Es brauchen lediglich alle die Teile des Programms weggelassen zu werden, die irgendwie mit dem PIO-Port A zu tun haben. Dieser Port wird in unserem Versuchs-Programm verwendet, um das Bitmuster eines ausgegebenen Bytes anzuzeigen.

In der Praxis erfolgt die serielle Datenübertragung ganz wesentlich schneller, als es in unserem Demonstrations-Programm geschieht. Ein Maß für die Übertragungs-Geschwindigkeit ist die Anzahl der in einer Sekunde übertragenen Bits. Diese Zahl wird mit **Baud** bezeichnet. Zu den in einer Sekunde übertragenen Bits gehören natürlich auch das Start-Ø-Bit und die beiden Stop-1-Bits. (Manchmal wird auch nur ein oder ein und ein halbes Stop-1-Bit übertragen.)

Eine oft gewählte Übertragungs-Geschwindigkeit ist 300 Baud, also 300 Bit/s. Da zur Übertragung eines Bytes einschließlich eines Start- und zweier Stop-Bits 10 Bit notwendig sind, ergibt das einen Datenfluß von 30 Byte/s. In der Datenverarbeitung sind aber auch wesentlich höhere Übertragungs-Geschwindigkeiten üblich, z. B. 9800 Baud und mehr.

Wenn unser Programm für eine Übertragungs-Geschwindigkeit von 300 Baud eingerichtet werden soll, dann erübrigt sich natürlich die im Bild S 65.1 dargestellte Reihenschaltung von zwei CTC-Kanälen. In unserer Darstellung kann der Kanal 1 wegfallen. Der Kanal Ø muß dann im *Timer Mode* arbeiten und vom System-Takt, also nicht mehr fremd über den CLK/TRG0-Anschluß, getriggert werden. In der INIT-Routine kann die Programmierung des CTC-Kanals 1 wegfallen; der CTC-Kanal muß entsprechend anders programmiert werden.

Nachdem Sie nun schon eine ganze Menge über den Aufbau und über die Funktion des Ein-/Ausgabebausteins 8255 gelesen haben, wird es Zeit, nun endlich ein erstes Programm zu schreiben. Dieses erste Programm soll zunächst einmal den Port C des Bausteins auf seine Ausgabefunktion in der Betriebsart 0 prüfen. Wichtig ist, daß immer zuerst das Steuerwort an den 8255 ausgegeben wird. Danach kann dann ein Datentransfer erfolgen.

Programmbeispiel S 67.1

Der 8255 soll zu Beginn so programmiert werden, daß der Port C des Bausteins auf Datenausgabe steht. Der Port A soll auf Dateneingabe und der Port B auf Datenausgabe programmiert werden.

Danach soll der Port C über die Leuchtdiodenreihe der Peripherie-Platine als 8-Bit-Binärzähler arbeiten und von ØØH bis FFH zählen. Die Zählgeschwindigkeit wählen wir mit ca. einer Sekunde. Ist der Zähler bei der sedezimalen Zahl FFH angelangt, soll das Programm stoppen.

Wir gehen ganz systematisch vor: Zuerst muß ein geeignetes Steuerwort gefunden werden. Dazu betrachten Sie bitte noch einmal den auf der Seite H 69 beschriebenen Aufbau des Steuerworts oder die Tabelle auf der Seite H 70. Sie werden schnell herausfinden, daß die Binärstruktur des Steuerworts wie folgt aussieht: 1ØØ1 ØØØØB. Diese Binärstruktur ist in der Sedezimal-Schreibweise die Zahl 9ØH. Wie bekommen wir nun diese Zahl in das Steuerwort-Register des Bausteins 8255?

Das ist nicht schwierig, denn Sie haben diesen Vorgang in ähnlicher Weise schon im Lehrbrief 1 bei der Programmierung der Z 80 PIO durchgearbeitet. Zuerst wird das gewünschte Steuerwort in den Akkumulator des Mikroprozessors geladen. Danach wird es durch den OUT-Befehl des Z 80 an das Steuerwortregister des 8255 ausgegeben, und schon ist die Initialisierung, so haben Sie die Programmierung des Steuerwortregisters auch bei der Z 80 PIO genannt, fertig. Das kleine Programmablaufdiagramm in Bild S 67.1 zeigt diesen Ablauf nochmals in anderer Darstellungsweise. Diesen Programmteil schreiben wir, wie schon gewohnt, als Unterprogramm mit der Bezeichnung INIT. Im Quellcode eines Assemblers würde dieses Unterprogramm wie folgt aussehen:

```
LD    A,9ØH        ;Steuerwort in den Akku laden
OUT   (C3H),A      ;Ausgabe an das Steuerwortregister
RET                ;Rückkehr zum Hauptprogramm
```

Damit sind nun alle Vorbereitungen getroffen, und das Hauptprogramm kann beginnen. Gefordert ist, daß der Port C des 8255 als Binärzähler programmiert werden soll. Das ist bestimmt mit Ihren Programmierkenntnissen keine Schwierigkeit mehr. Bild S 68.1 zeigt den prinzipiellen Ablauf des Programms.

Bild S 67.1
Ein recht kurzes Unterprogramm, das lediglich den 8255 initialisiert. Das bedeutet nichts weiter, als daß die Funktion der Ports festgelegt wird.

Der zweite Abschnitt des Programms ist das Definieren eines Zählwerks im Mikroprozessor. Das Einfachste ist, wenn Sie den Akkumulator auch als Zählwerk benutzen. Selbstverständlich kann auch irgend ein Register oder eine RAM-Speicherstelle als Zähler definiert werden. Bleiben wir beim Akkumulator. Der Anfangsstand des Zählwerks soll

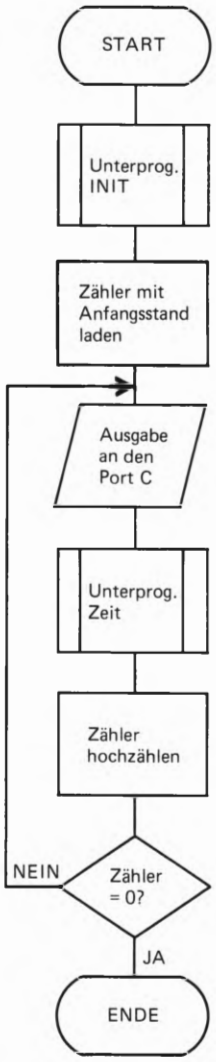


Bild S68.1
Der Programmablaufplan für das Hauptprogramm. Der Zähler wird solange hochgezählt, bis der Wert FFH überschritten ist. Danach ist das Programm zu Ende.

natürlich ØØH sein. Dementsprechend wird zu Beginn diese Zahl auch in den Akkumulator geladen. Mit dem OUT-Befehl kann dann dieser Zahlenwert an den Port C des 8255 weitergeleitet werden. Laut Programmablaufplan folgt danach eine Sekunde Pause. Die „Warte-Routine" wird als Unterprogramm mit der Bezeichnung „Zeit" geschrieben.

Nun folgt die Abfrage nach dem Zahlenwert im Akkumulator. Ist der Stand des Akkumulators FFH, dann ist das Zählwerk bei seinem Höchstwert angekommen und somit das Programm fertig. Ist dies nicht der Fall, muß der Akkumulator hochgezählt und die Zahl an den Port C weitergegeben werden.

Im Quellcode sieht dieser Programmteil wie folgt aus:

```
;Hauptprogramm

START:  LD    A,ØØH          ;Anfangsstand des Zählers laden
LOOP:   OUT   (PORT C),A     ;Ausgabe an den Port C
        CALL  ZEIT           ;Unterprogramm Zeit aufrufen
        INC   A              ;Zähler hochzählen
        JR    NZ,LOOP        ;Abfrage auf Programmende
        HALT                 ;Programmende
```

Das Unterprogramm Zeit

Das Unterprogramm Zeit bedarf keiner besonderen Beschreibung. Die Zeitverzögerung wird durch einfaches Decrementieren zweier Register erreicht. Für diese Tätigkeit kann man jedes beliebige Register heranziehen. In unserem Programmteil haben wir das Register H und das Register L verwendet. Damit dieser Programmteil auch universal einsetzbar wird, rettet man diese beiden Register zu Beginn des Programmteils auf den Stack, und holt die Inhalte am Ende des Unterprogramms von dort wieder ab. Wenn Sie ganz sicher sein wollen, daß das Unterprogramm keine Registerinhalte zerstört, die eventuell im Hauptprogramm wieder benötigt werden, dann sollte auch der Inhalt des Flagregisters gerettet werden. Dazu muß in diesem Fall der Akku und das Flagregister durch PUSH AF auf den Stack gebracht werden. Der Quellcode des Programms lautet nun:

```
;Unterprogramm ZEIT

ZEIT:   PUSH  HL             ;HL in den Stack retten
        PUSH  AF             ;AF in den Stack retten
        LD    H,ØFFH         ;Zähler 1 laden
LOOP1:  LD    L,ØFFH         ;Zähler 2 laden
LOOP2:  DEC   L              ;Zähler 2 decrementieren und
        JR    NZ,LOOP2       ;auf Null abfragen
        DEC   H              ;Zähler 1 decrementieren und
        JR    NZ,LOOP1       ;auf Null abfragen
        POP   AF             ;AF aus dem Stack zurückholen
        POP   HL             ;HL aus dem Stack zurückholen
        RET                  ;Rücksprung ins Hauptprogramm
```

Versuch S69.1

Binärer Zähler über PORT C

Das Programm ist nun fertig. Das vollständige Programmlisting finden Sie auf der Seite L67. Nun bedarf es nur noch der Eingabe des Programms in das Mikroprozessorsystem. Jetzt ist also „Handarbeit" gefragt! Aber das haben Sie ja schon so oft gemacht, daß es bestimmt keiner Erklärung mehr bedarf.

Außerdem muß natürlich die Peripherie-Platine an den „Micro-Professor" angeschlossen werden. Das ist jedoch keine Schwierigkeit. Sie müssen lediglich darauf achten, daß Sie das Flachbandkabel, mit dem Sie die Peripherie-Platine in den Lehrbriefen 1 bis 3 angeschlossen hatten, nun entfernen. Leider kann die Z 80 PIO nicht mit dem 8255 gleichzeitig betrieben werden. Das kommt einfach daher, daß die PORTs der beiden Bausteine parallel auf der Peripherie-Platine verdrahtet sind und sich gegenseitig beeinflussen. Das würde aber zu mißverständlichen Ausgangssignalen führen. Also zuerst das Flachbandkabel für die PIO entfernen und dann erst das obere Flachbandkabel, so wie es das Bild S69.1 zeigt, einstecken.

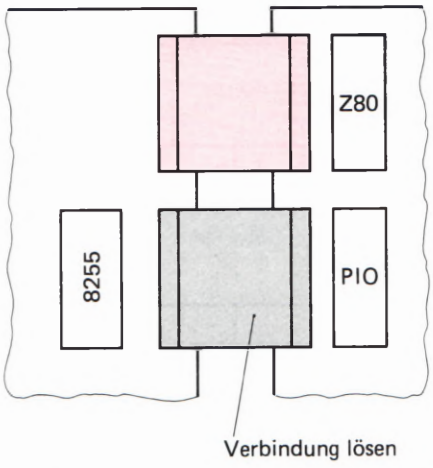


Bild S69.1
Das obere (rot gekennzeichnete) Flachbandkabel verbindet den Adreß-, Daten- und Steuerbus des Micro-Professors mit der Peripherie-Platine. Bitte beachten Sie, daß jeweils nur ein Flachbandkabel eingesteckt werden darf, weil sonst die Z 80 PIO und der 8255 gleichzeitig an dieselbe Leuchtdiodenreihe angeschlossen sind.

Die Stiftleiste für die steckbaren Brücken können Sie momentan außer acht lassen, weil das Programm ausschließlich Ausgaben macht – also keinen Einlesevorgang hat. Am übersichtlichsten ist es, wenn Sie alle steckbaren Brücken herausnehmen. Im nächsten Programm werden Sie diese Brücken wieder benutzen.

Wenn Sie das Programm eingetastet und gestartet haben, muß am PORT C der binäre Zähler zu beobachten sein.

Lauflicht mit Ausgabe auf Port A

Sie kennen bestimmt sogenannte Lauflichter von Straßenbaustellen her. Dort werden Reihen von einzelnen Lampen nacheinander sehr kurz angesteuert, um Baustellen in der Nacht gut erkenntlich zu machen. So eine Art Lauflicht läßt sich natürlich recht einfach mit einem Mikroprozessor-System programmieren. Und zur Darstellung des Lauflichts verwenden wir einfach eine Leuchtdiodenreihe auf der Peripherie-Platine.

Der Aufbau des Programms ist denkbar einfach. Zu Beginn des Programms wird erst einmal der 8255 initialisiert. Das erledigen wir wieder innerhalb eines Unterprogramms und nennen es wie im vorigen Beispielprogramm „INIT".

Das Lauflicht selbst wird erzeugt durch die Ausgabe verschiedener Bit-Muster, die wie folgt aussehen:

```
0000 0000
0000 0001
0000 0010
0000 0100
0000 1000
0001 0000

1000 0000
```

Zwischen der Ausgabe der einzelnen Bit-Muster muß jeweils eine kurze Zeitverzögerung eingebaut sein. Sind alle Bit-Muster durchlaufen, soll das Programm eine einstellbare Warteschleife durchlaufen, die über den Port C variiert werden kann.

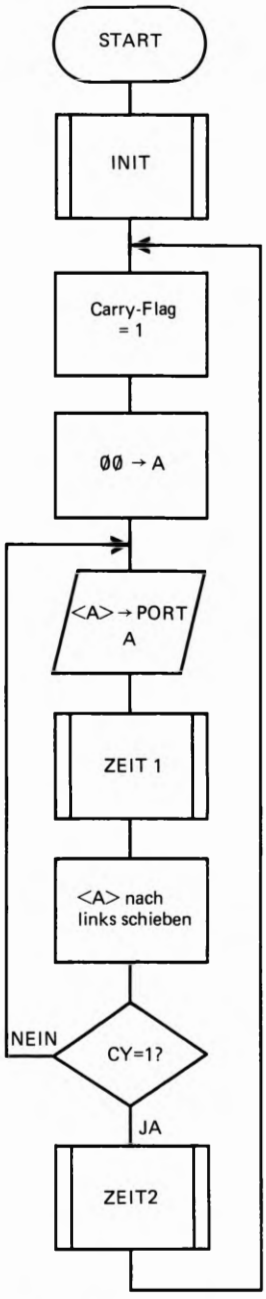


Bild S70.1
Der Programmablaufplan des Hauptprogramms. Durch den Schiebevorgang wird ein Lauflicht erzeugt.

Die Struktur des Programmaufbaus ist recht einfach. Zuerst wird der 8255 vorprogrammiert. Der nächste Programmschritt ist, die Anfangsbedingungen für die erste Ausgabe auf den Port A zu schaffen. Es gibt bestimmt verschiedene programmtechnische Möglichkeiten, das Lauflicht zu realisieren. Wir haben es in diesem Programm so gemacht, daß zuerst das Carry-Flag mit einer Eins und der Akku mit Nullen geladen wird. Das ist schon die ganze Anfangsbedingung. Im weiteren Programmablauf gibt der OUT-Befehl den Akkuinhalt an den Port A des 8255 weiter. Ist dies geschehen, braucht der Akkuinhalt lediglich um eine Stelle nach links verschoben zu werden, und zwar „durch das Carry-Bit". So kommt die Eins, die zunächst im Carry-Bit stand, in den Akkumulator und wird darin in jedem Programmdurchlauf um eine Stelle weiter nach links geschoben. Bild B 70.2 zeigt diesen Schiebevorgang im Akkumulator.

Zwischen jedem erneuten Schiebevorgang im Akkumulator muß eine kurze Zeitverzögerung durchlaufen werden. Um das Programm so übersichtlich wie möglich zu machen, ist dieser Programmteil als Unterprogramm geschrieben, mit der Bezeichnung ZEIT 1. Bitte sehen Sie sich den Programmablaufplan in Bild S 70.1 an.

Sind alle Leuchtdioden des Ports A einmal angesteuert worden, beginnt der Schiebevorgang erneut. Zuvor soll jedoch eine größere Zeitverzögerung stattfinden. Dies erreicht man durch das Unterprogramm ZEIT 2. Wie dieses Zeitverzögerungsprogramm variabel wird, sehen wir uns gleich etwas näher an. Zuvor jedoch noch ein paar Worte zum Initialisierungsprogramm.

Initialisierungsprogramm INIT

Zwei Forderungen muß das Unterprogramm INIT erfüllen: Port A muß so programmiert sein, daß er als Ausgabeport arbeitet. Über den Port C wird eine Zeitkonstante eingelesen, die für die Länge der Zeitverzögerung des Unterprogramms ZEIT 2 zuständig ist. Dieser Port muß demnach als Eingabeport arbeiten.

Bleibt noch der Port B des 8255 übrig. Da dieser auf Ihrer Peripherie-Platine nicht direkt zugänglich ist, spielt es auch keine Rolle, ob er auf Aus- oder auf Eingabe programmiert wird. Der Einfachheit halber wird er auf Ausgabe programmiert; so werden alle Bits des Steuerworts, die für den Port B zuständig sind, mit einer Null belegt.

Das Steuerwort für die Initialisierung ist damit:

10001001

Betrachten Sie zur Überprüfung des Steuerworts noch einmal die Darstellung in Bild H 69.1. Das Bit DØ des Steuerworts muß demnach eine logische 1 aufweisen, denn der Port C soll ja als Eingabeport arbeiten. Port B wird durch die Ø an Bit D1 zum Ausgabeport. D2 entscheidet, in welcher Betriebsart der Port B und der Port C arbeiten sollen. Für den nicht verwendeten Port B ist dies bedeutungslos, weil er nicht verwendet wird. Port C muß jedoch in der Betriebsart 0 arbeiten, dementsprechend wird dieses Bit zu einer logischen Ø.

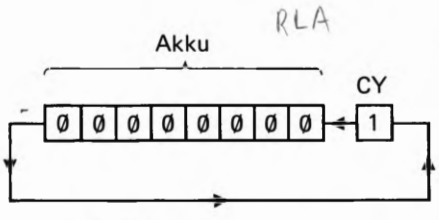


Bild S70.2
So wird das Carry-Bit durch den Akku hindurch geschoben. Eine logische 1 bedeutet, daß die eine Leuchtdiode am Port A aufleuchtet.

Das Bit D3 des Steuerworts entscheidet über die Funktion des Ports C, und zwar über die höherwertigen vier Bits. Diese müssen, wie die niederwertigen 4 Bits dieses Ports, als Eingabeport arbeiten, und somit ist eine 1 an dieser Stelle des Steuerworts notwendig.

Nun folgt durch die nächsten drei Bits die Programmierung des Ports A. Gefordert ist, daß Port A als Ausgabeport in der Betriebsart 0 arbeitet. Wenn Sie Bild H69.1 betrachten, sehen Sie, daß hierfür D4, D5 und D6 eine logische Ø bekommen müssen. Bleibt also nur noch das Betriebsarten-Kennzeichenbit, das vorerst immer eine Eins haben muß. Damit sind nun alle Ports programmiert. Wenn Sie nun noch die Binärstruktur in eine Sedezimalzahl umwandeln, werden Sie für das Steuerwort des 8255 die Zahl 8BH erhalten.

Das Unterprogramm INIT ist damit auch schon fast festgelegt. Die Zahl 8BH wird in den Akkumulator geladen und wie im vorigen Beispielprogramm durch den OUT-Befehl an das Steuerwortregister im 8255 weitergegeben. Die gewünschte Funktion bleibt dann solange erhalten, bis ein neues Steuerwort an das Steuerwortregister weitergegeben wird oder ein RESET-Signal erfolgt.

Das variable Zeitprogramm ZEIT 2

Eine Variante eines Zeitverzögerungsprogramms haben Sie schon im letzten Programmbeispiel kennengelernt. Der Nachteil dieses Programms war, daß sich immer die gleiche Zeitverzögerung einstellte. Der Binärzähler lief somit immmer mit der gleichen Geschwindigkeit. Für manche Anwendungen ist es jedoch von großem Vorteil, wenn durch eine Schalterkombination die Zeitverzögerung eingestellt werden kann. Das Zeitprogramm ZEIT 2 soll diese Funktion erfüllen. Bild S71.1 zeigt den Programmablaufplan des Zeitverzögerungsprogramms.

Zuerst liest das Programm den an Port C des 8255 anstehenden Zahlenwert in den Akkumulator ein. Dies geschieht ganz einfach durch einen IN-Befehl. Der Port C des 8255 wird im Initialisierungsprogramm auf die Betriebsart 0 programmiert. Das bedeutet, daß keine Steuersignale oder Quittungssignale den genauen Zeitpunkt des Einlesevorgangs festlegen. Wann genau der Zahlenwert durch den IN-Befehl eingelesen wird, ist nicht bestimmt vorauszusagen. Wir wissen lediglich, daß der Einlesevorgang dann erfolgt, wenn im Hauptprogramm alle Leuchtdioden einmal angesprochen wurden. Den Zahlenwert durch die Taster der Peripherieplatine einzustellen, ist nicht sinnvoll. Besser ist es, die parallel zu den Tastern angeordneten Schalter zu verwenden, damit stehen die einzulesenden Daten sozusagen immer am Port C an. Wann sich das Programm die Daten abholt, spielt keine Rolle. Durch die Stellung der Schalter können Sie Zahlenwerte zwischen 0 und 255 einstellen und erreichen damit ganz verschiedene Zeitverzögerungen.

Das Programm ZEIT 2 verwendet das L-Register des Prozessors als Zähler. Wenn Sie sich den Programmablaufplan in Bild S71.1 betrachten, sehen Sie, daß der von Port C eingelesene Zahlenwert in das L-Register übertragen wird. Danach ruft das Programm das Unterprogramm ZEIT 1 so oft auf, bis das L-Register auf Null gezählt ist. Wie oft dies geschieht, entscheidet der eingelesene Zahlenwert. Sie sehen, auch dieser Programmteil ist nicht schwierig zu programmieren.

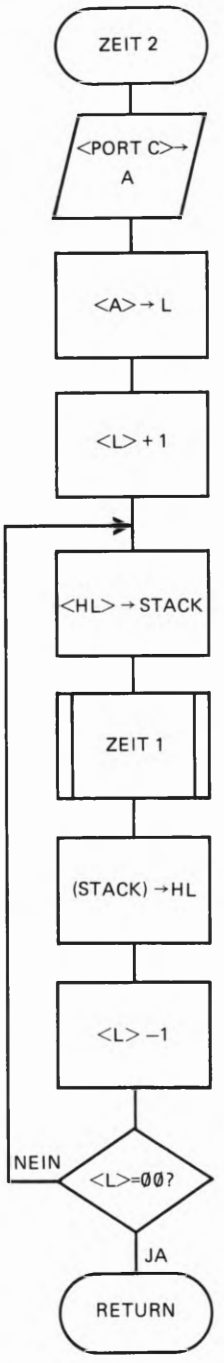


Bild S71.1
Über den Port C des 8255 wird die Zeitkonstante eingelesen. Die Zeitkonstante kann an der Schalterreihe auf der Peripherie-Platine eingestellt werden.

S
72

Versuch S 72.1

Verbinden Sie die Micro-Professor-Platine mit der Peripherie-Platine so, wie es Bild S 69.1 zeigt. Die steckbaren Brücken müssen auf der mit B/C gekennzeichneten Leiste stecken. Im Programm sollen ja Eingaben über den Port C gemacht werden. Port A benötigt keine Brükken, er arbeitet nur als Ausgabeport. Damit sind auch schon alle Hardware-Vorbereitungen getroffen, um das eben besprochene Programm auszutesten. Das vollständige Programm finden Sie auf der mit der Griffmarke L 68 gekennzeichneten Seite dieses Lehrbriefs.

Tasten Sie das Programm einmal ein, und starten Sie es mit der Adresse 1800H. Die Leuchtdioden blitzen danach von rechts nach links kurz auf. Je nachdem welche Zahl Sie an den Schaltern eingestellt haben, ist die Zeitverzögerung zwischen zwei Schiebedurchläufen unterschiedlich lang.

Aufgabe S 72.1

Betrachten Sie noch einmal den Programmablaufplan in Bild S 71.1. Sie sehen, daß nach dem Einlesevorgang das L-Register um den Zahlenwert 1 hochgezählt wird. Warum ist dieser Vorgang in diesem Unterprogramm notwendig. Oder kann er eventuell ganz weggelassen werden und das Programm arbeitet immer noch einwandfrei?

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 11.

Programmierung des 8255 auf der Micro-Professor-Platine

Wie der 8255 an den Adreß-, Daten- und Steuerbus des Micro-Professor-Entwicklungssystems angeschlossen ist, haben Sie nun schon aus dem vorigen Kapitel Hardware kennengelernt. Wichtig ist, daß Sie sich die Adressierung der verschiedenen Ports und des Steuerwortregisters dieses Bausteins gut merken. Am einfachsten ist es, wenn wir kurz ein ganz simples Programm zur Aktivierung der Anzeige auf dem Entwicklungssystem schreiben.

Funktionsbeschreibung

Das Programm wird die erste Stelle (rechts außen) der sechsstelligen Anzeigeeinheit aktivieren. Danach soll eine '1' an dieser Stelle der Anzeigeeinheit aufleuchten. Ein recht einfaches Programm also, das aber die Aktivierung einer Anzeigestelle durchaus einleuchtend demonstriert.

Die Programmentwicklung ist schnell gemacht. Zu aller erst muß der 8255 entsprechend vorprogrammiert werden. Dazu sehen Sie sich bitte noch einmal das Bild H72.1 an. Der Port B und der Port C des Bausteins bedienen die Anzeigeeinheit. Es werden auf beiden Ports Daten ausgegeben, und zwar ohne irgendwelche Quittungssignale. Das bedeutet, daß sowohl die Betriebsart als auch die Datenflußrichtung für Port B und Port C festliegen.

Beide Ports arbeiten in der Betriebsart 0 und als Ausgabeports. Was ist aber mit dem Port A? Nun, diesen Port benutzen wir im folgenden Programm nicht, und deshalb ist es eigentlich egal, ob er als Ein- oder als Ausgabeport arbeitet. Für die Abfrage der Tastatur, das sei hier vorweggenommen, muß dieser Port als Eingabeport arbeiten; ebenfalls in der Betriebsart 0. Das gewünschte Steuerwort für den 8255 ist entsprechend Bild H69.1 die Sedezimalcodierung 9ØH.

Sobald das Steuerwort an das Steuerwortregister durch einen OUT-Befehl ausgegeben wird, kann die Aktivierung der rechts außen stehenden Anzeige erfolgen. Zu diesem Zweck wird das folgende Bitmuster an den Port C ausgegeben: ØØØØ ØØØ1B. Dieses Bitmuster bewirkt, daß nur die gewünschte Anzeige aktiviert ist, alle anderen sind inaktiv.

Im dritten Programmschritt wird nun ausgewählt, welche Segmente der aktivierten Anzeige aufleuchten sollen. Wir haben vereinbart, daß eine 1 aufleuchten soll. Das bedeutet, daß die mit b und c bezeichneten Segmente angesprochen werden müssen. Bitte betrachten Sie nun noch einmal das auf der Seite H73 dargestellte Schema. Daraus geht hervor, welches Bitmuster notwendig ist, um die Segmente b und c leuchten zu lassen. Sie werden feststellen, daß es das Bitmuster ØØ11 ØØØØB sein muß. Der Port B ist sozusagen für die einzelnen Segmente einer Anzeige zuständig. Es gilt also, nun das Bitmuster ØØ11 ØØØØB an diesen Port auszugeben. Damit ist unser kleines Anzeigeprogramm auch schon fertig. Der Programmablaufplan in Bild S73.1 gibt Ihnen eine Übersicht über dieses Programm.

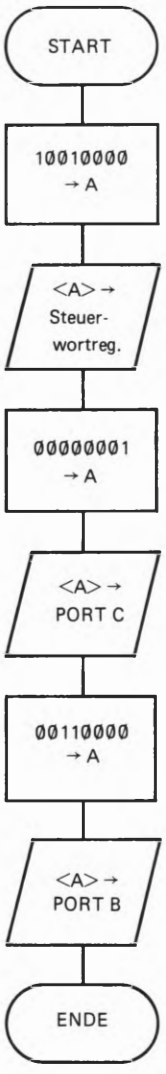


Bild S73.1
Der Programmablaufplan für die Aktivierung der Anzeigeeinheit. Es wird die rechts außen stehende Anzeige aktiviert und an dieser Stelle eine 1 angezeigt.

Aufgabe S 74.1

Setzen Sie bitte den Programmablaufplan für das eben besprochene kleine Anzeigeprogramm einmal selbst in ein lauffähiges Programm um. Beginnen Sie das Programm mit der Anfangsadresse 18ØØH.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 11.

Ein kleiner Fehler mit großer Wirkung

Wenn Sie das Programm entwickelt und in das Micro-Professor-Entwicklungssystem eingetastet haben, werden Sie zunächst eine kleine Enttäuschung erleben. Das Programm scheint die gewünschte Funktion nicht zu erfüllen. Das ist bei Programmentwicklungen meist ganz normal. Nun beginnt die Fehlersuche. Bestimmt gehen Sie wie wir auch so vor, daß zuerst mit grimmiger Miene das eingegebene Programm auf eventuelle Tippfehler untersucht wird.

In diesem Programm werden Sie wahrscheinlich keinen Tippfehler gefunden haben. Also geht man zum nächsten Schritt über. Irgendwo muß ein Fehler im Programm zu finden sein. Zunächst einmal zurück zu dem, was das Programm im Moment tut. Starten Sie das Programm. Wenn Sie dies bei der Adresse 18ØØH tun, läuft das Programm bis zur Adresse 18ØDH und bleibt dann stehen. Angezeigt wird die Adresse und deren Inhalt. Das ist aber nicht das, was wir ursprünglich erwartet haben.

Eigentlich sollte eine 1 in der rechten Anzeige erscheinen, alle anderen Anzeigen sollten dunkel bleiben. Ein direkter Fehler kann bei diesem kleinen Programm im Grunde gar nicht vorkommen, das haben Sie sich bestimmt auch schon gedacht. Vielleicht hilft ein Blick auf den Schaltplan in Bild H 72.1. Wenn Sie einmal genau hinsehen, werden Sie bemerken, daß wir die Bits PC6 und PC7 durch die ausgewählte Bitstruktur im Programm einfach nicht berücksichtigt haben. Wir haben diese beiden Bits jeweils mit einer Ø belegt und gar nicht bedacht, daß manche Signale auch mit einer Ø aktiv werden. Genau das ist uns aber bei der Auswahl des Bitmusters passiert. Das Bit 6 des Ports C hat nämlich eine besondere Bedeutung: Es löst einen nicht maskierbaren Interrupt am Z 80 aus. Dieser bewirkt, daß der Programmzähler mit der Adresse ØØ66H belegt wird. Das wiederum bedeutet, daß das Programm unterbrochen wird und im Monitorprogramm des Micro-Professors weiterarbeitet. Das war aber nicht unsere Absicht.

Der kleine Fehler mit der großen Wirkung ist also schnell behoben. Wir berücksichtigen einfach bei der Auswahl des Bitmusters, daß das Bit 6 von Port C immer eine 1 aufweisen muß, weil sonst ein ungewollter Interrupt stattfindet. Das neue Bitmuster lautet dementsprechend: Ø1ØØ ØØØ1B.

Ändern Sie bitte das Bitmuster im Programm. Sie werden sehen, das Programm läuft nun wunschgemäß ab. Ganz rechts außen in der Anzeigereihe leuchtet eine 1 auf – so, wie vereinbart. Den kleinen Fehler haben wir ganz bewußt in das Programm eingebaut, damit Sie sehen, wie abhängig die Software von der Hardware ist und umgekehrt. Da wir durch den Fehler im Bitmuster mehr oder minder durch

Zufall auf einen ungewollten Interrupt gestoßen sind, soll an dieser Stelle des Lehrgangs ein weiterer Interrupt besprochen werden. Wiederholen wir einmal ganz kurz, was an verschiedenen Stellen der ersten drei Lehrbriefe des Lehrgangs bereits über einen Interrupt gesagt wurde.

Interrupt mit Restart RST 38

Ganz allgemein gilt, ein Interrupt wird in der Regel direkt durch eine Steuerleitung, die zum Mikroprozessor führt, angefordert. Für diesen Zweck hat der Z 80 zwei verschiedene Interrupt-Anforderungsleitungen. Es sind dies die Steuerleitungen mit den Bezeichnungen $\overline{\text{NMI}}$ und $\overline{\text{INT}}$. Beide Leitungen werden mit einem Ø-Signal aktiv.

$\overline{\text{NMI}}$ steht für die Bezeichnung „nicht maskierbarer Interrupt". Dahinter verbirgt sich nichts Geheimnisvolles. Es bedeutet lediglich, daß auf einen angeforderten Interrupt immer die gleiche Reaktion erfolgt: Der Programmzähler wird mit der Adresse ØØ66H geladen. In dieser Adresse steht dann in der Regel, wo sich die Interrupt-Service-Routine befindet. Beim Micro-Professor-System hat man die Service-Routine ab der Adresse ØØ66H geschrieben.

Wird ein Interrupt über die Steuerleitung mit der Bezeichnung $\overline{\text{INT}}$ angefordert, gibt es mehrere Möglichkeiten, wo die Interrupt-Service-Routine abgelegt sein kann. Aus der Sicht des Mikroprozessors läuft nach einer Interrupt-Anforderung folgendes ab:

1. Das Interrupt-Freigabe-Flipflop wird abgefragt, ob eine Programmunterbrechung überhaupt gestattet ist. Ist dies nicht der Fall, dann wird die Interrupt-Anforderung ignoriert. Im anderen Fall wird dem Interrupt stattgegeben. Ein Interrupt wird durch den Befehl EI freigegeben und durch den Befehl DI gesperrt. Das wissen Sie aber bestimmt schon aus der Interrupt-Behandlung in den vorigen Lehrbriefen. Gehen wir also gleich zum nächsten Schritt über.
2. Der Programmzählerstand, bei dem das Hauptprogramm durch den angeforderten Interrupt unterbrochen wurde, wird auf den Stack gerettet. Dies ist notwendig, weil der Mikroprozessor nach Abarbeiten der Interrupt-Service-Routine wissen muß, an welcher Stelle des Programms weitergearbeitet wird.
3. Der jetzt folgende Arbeitsablauf ist fast der wichtigste und sehr bedeutungsvoll für die Hardware eines Mikroprozessorsystems. Wie vor jedem neuen Programmschritt liest nun der Mikroprozessor einen neuen Befehl, der auf dem Datenbus anstehen muß, ein. Bei der Bearbeitung eines Interrupts ist der anstehende Befehl einer der sogenannten Restart-Befehle. Dieser kommt allerdings nicht wie ein „normaler" Befehl aus dem Speicher des Mikroprozessorsystems, sondern muß von der Hardware des Systems sozusagen extern auf den Datenbus aufgeschaltet werden. Bei der Verwendung der Z 80 PIO erledigt die PIO diese Aufgabe selbst. Wenn aber ein 8255 als Ein-Ausgabe-Baustein verwendet wird, muß der Restart-Befehl durch eine zusätzliche Hardware auf den Datenbus aufgeschaltet werden.

Das Micro-Professor-System hat hierfür keine zusätzliche Hardware zur Verfügung – zumindest nicht auf den ersten Blick. Zunächst sieht es so aus, als ob Interrupts in Zusammenhang mit dem 8255 gar nicht zu verarbeiten wären. Nun, bei genauerer Betrachtung der Schaltung ergibt sich doch noch eine Möglichkeit, zumindest mit einem Restart-Befehl arbeiten zu können. Alle Leitungen des Datenbus sind durch Widerstände auf 5-Volt „hochgezogen". Diese Tatsache hilft uns weiter, weil einer der Restart-Befehle die Sedezimalcodierung FFH hat. Genau dieses Bitmuster liegt vor, wenn der Restart-Befehl mit der Bezeichnung RST 38 aufgerufen werden soll. Was geschieht nun bei diesem Befehl?

4. Der RST 38-Befehl hat zur Folge, daß der Programmzähler mit der Adresse ØØ38H geladen wird. Das können Sie selbst ausprobieren, indem Sie diesen Befehl im Schrittbetrieb austesten. Geben Sie einmal folgende Programmzeile ein: 18ØØ FF. Wenn Sie diese Programmzeile durch STEP starten, werden Sie beobachten, daß das Programm zur Adresse ØØ38H verzweigt wird. Beginnend mit dieser Adresse könnte jetzt der Anwender seine Interrupt-Service-Routine ablegen. Bestimmt wissen Sie, daß der Speicherbereich an dieser Stelle des Micro-Professor-Systems ROMs aufweist. Hier kann also der Anwender nichts selbst programmieren. Aus diesem Grunde haben die Entwickler des Micro-Professors eine kleine Programmroutine, beginnend mit der Adresse ØØ38H, in das System eingebaut. Diese Routine hat den Zweck, das Programm so zu verzweigen, daß der Anwender selbst bestimmen kann, wo seine Interrupt-Service-Routine beginnen soll. Entscheidend sind die Inhalte der Adressen 1FEEH und 1FEFH. In diese beiden Adressen kann der Anwender die Anfangsadresse seiner Interrupt-Service-Routine ablegen. Wie das im Einzelnen gemacht wird, sehen Sie im gleich folgenden Programm.

5. Nehmen wir einmal an, nun sei die Interrupt-Service-Routine abgearbeitet. Was darin im Einzelnen geschieht, soll im Moment nicht interessieren. Sobald die Routine abgearbeitet ist, holt sich der Mikroprozessor die Rücksprungadresse vom Stack-Bereich ab und arbeitet genau dort weiter, wo er das Programm vor der Abarbeitung des RST-Befehls verlassen hat.

So, das war bestimmt etwas viel Theorie aufeinmal. Das einfachste Gegenmittel ist immer ein Programm zu schreiben. Damit werden die notwendigen Programmschritte bei der Bearbeitung eines Interrupts bestimmt deutlicher.

Bild S76.1
Im Hauptprogramm wird die Anzeige aktiviert und der jeweilige Zählerstand in der Anzeige dargestellt.

Programmbeispiel S76.1

Das folgende Programmbeispiel soll nichts weiter tun, als die Anzahl der eingetroffenen Interrupt-Anforderungen zu zählen und in der rechten Anzeige darzustellen. Das Hauptprogramm ist grundsätzlich mit dem vorigen Programm auf der Seite S73 identisch, wir haben nur kleine Änderungen vorgenommen. Der Programmablaufplan in Bild S76.1 zeigt die Struktur dieses Programms. Daraus können Sie erkennen, daß gleich zu Anfang des Programms die Speicherstellen 1FEEH

und 1FEFH mit der Sedezimalzahl 19ØØH geladen werden. Das bedeutet, die Interrupt-Service-Routine muß für unser Programmbeispiel bei der Adresse 19ØØH beginnen. Im nächsten Programmschritt wird der Interrupt-Ereigniszähler mit der Zahl Null geladen. Das ist sozusagen eine Anfangsbedingung. Als Zähler kann grundsätzlich jedes Register des Mikroprozessors verwendet werden. Wir haben für diesen Zweck das E-Register gewählt.

Der Initialisierung des 8255 brauchen wir keine langen Ausführungen zu widmen, sie ist genau gleich wie die Initialisierung des Programms auf Seite S73. Auch die Aktivierung der Anzeige ist Ihnen nun schon bekannt. Es wird einfach das Bitmuster Ø1ØØ ØØØ1B an den Port C des 8255 ausgegeben.

Die folgenden Programmschritte sind aber neu. Wir können ja nicht jeden neuen Zählerstand während eines Programmablaufs im Kopf in den 7-Segment-Code umwandeln. Das muß jetzt das Programm selbst tun. Selbstverständlich könnten Sie mit Ihren Programmierkenntnissen ein kleines Programm selbst schreiben, das für die jeweilige Zahl die entsprechenden Segmente der Anzeige aktiviert. Wir ersparen uns aber diese Arbeit, indem wir hierfür ein schon im Monitorprogramm des Micro-Professors vorhandene Umwandlungsroutine verwenden. Sie trägt den Namen „HEX 7" und wandelt eine im Akkumulator stehende Zahl in den 7-Segment-Code um. Die Anfangsadresse des Unterprogramms ist Ø689H.

Sobald der Zählerstand aus dem E-Register im Akku angelangt ist, kann das Unterprogramm HEX 7 aufgerufen werden. Danach ist im Akku das erforderliche Bitmuster zur Aktivierung der einzelnen Segmente der Anzeige zu finden. Der vorige Akku-Inhalt ist allerdings überschrieben worden, und der Zustand der Flags hat sich geändert. Das sollten Sie bei der Verwendung dieses Unterprogramms immer bedenken.

Nun ist unser Hauptprogramm einmal durchlaufen worden. Es wird weiterhin der Zählerstand abgefragt und angezeigt. Dies geschieht endlos, und zwar so lange, bis ein Interrupt angefordert wird. Den Interrupt können Sie durch die mit INTR bezeichnete Taste des Micro-Professor-Systems anfordern. Diese Taste finden Sie in der ganz links außen stehenden Tastenreihe.

Die Interrupt-Service-Routine

Die Struktur der Interrupt-Service-Routine sehen Sie in Bild S77.1. Was beinhaltet diese Programmroutine?

Zu allererst wird eine eventuell eintreffende erneute Interrupt-Anforderung gesperrt. Dies erreicht man mit dem Befehl DI. Damit sperrt das Interrupt-Flipflop neue Interruptsignale solange, bis per Software dieses Steuersignal wieder freigegeben wird. Der zweite Programmschritt ist der Aufruf eines Zeitverzögerungsprogramms. Dieser Programmschritt wird notwendig, weil die Interrupttaste des Micro-Professor-Systems prellt.

Bild S77.1
Der Programmablaufplan der Interrupt-Service-Routine. Das eingetragene Zeitverzögerungsprogramm ist nicht extra aufgeführt. Es ist mit dem auf Seite S68 gezeigten Unterprogramm ZEIT nahezu identisch.

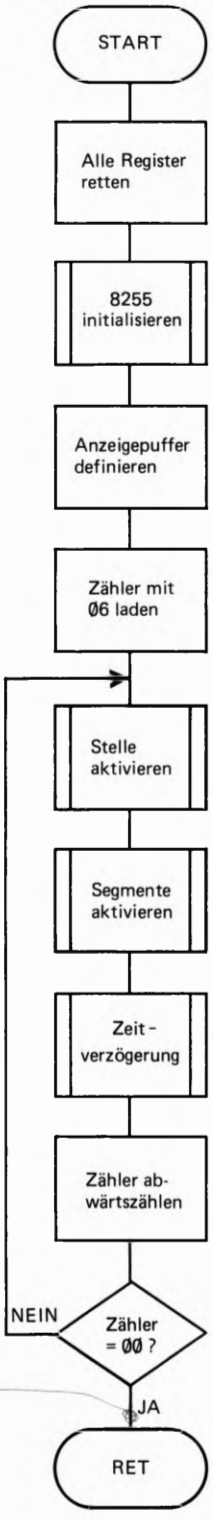


Bild S78.1
Der Programmablaufplan zur Ansteuerung der Anzeige. Programmstart ist die Adresse 1800H.

Danach erhöht das Programm den Zählerstand um 1 und vergleicht den Zählerstand mit der Zahl 0AH. Der Vergleich wurde aus einem einfachen Grund in das Programm eingebaut: Wir haben nur eine Anzeige zur Verfügung. Deshalb sollte die angezeigte Zählerzahl nur einstellig sein. Das Programm zählt somit nur 9 Interrupt-Anforderungen. Danach wird der Zählerstand 00 wieder eingestellt. Der letzte Programmschritt gibt erneute Interrupts frei und läßt das Programm an die Stelle zurückkehren, wo es einmal unterbrochen wurde. Wichtig ist, daß die Interrupt-Service-Routine mit dem Befehl RETI abgeschlossen wird. Damit ist unsere Interrupt-Service-Routine zu Ende.

PER14 S78/EDI L69

Versuch S78.1

Das fertige Programm finden Sie auf der Seite L69. Tasten Sie dieses Programm in das Micro-Professor-Entwicklungssystem ein. Wenn Sie das Programm mit der Adresse 1800H starten, muß eine Null in der rechten Anzeige aufleuchten. Sobald Sie die Interrupttaste betätigen wird sich die angezeigte Zahl erhöhen.

Ersetzen Sie als kleinen Test den Aufruf des Zeitverzögerungsprogramms mit drei NOP-Befehlen. Wenn Sie danach das Programm erneut starten, sehen Sie anhand des Zählerstandes, daß die Taste des Systems prellt. Man muß die Taste schon extrem kurz drücken, wenn der Zähler wie gewünscht arbeiten soll.

Vollständige Bedienung der Anzeigeeinheit

In den beiden letzten Programmen haben Sie kennengelernt, wie der 8255 auf der Micro-Professor-Platine arbeitet. Außerdem haben Sie gesehen, wie eine Anzeige aktiviert und deren Segmente angesteuert werden. Das nun folgende Programm geht einen Schritt weiter. Es zeigt, wie alle sechs 7-Segment-Anzeigen nacheinander angesteuert werden.

Der Anzeigepuffer

Im letzten Programm haben wir das E-Register als Zähler benutzt, dessen Zählerstand angezeigt wurde. Das E-Register könnte man in dieser Funktion auch „Anzeigepuffer" nennen. Will man in alle sechs Anzeigen nacheinander verschiedene Daten anzeigen, empfiehlt es sich, einen sechsstelligen Anzeigepuffer zu definieren. Was heißt das? Nun, sechs Register für die Anzeige der sechs Stellen zu reservieren, wäre eine unnütze Verschwendung der Register des Mikroprozessors. Eine andere Möglichkeit zur Lösung des Problems ist, einfach sechs RAM-Speicherplätze als Anzeigepuffer zu benutzen. Wir haben willkürlich den Speicherbereich 1900H bis 1905H definiert. Das folgende Schema zeigt die Bedeutung dieses Speicherbereichs.

Speicherbereich	Bedeutung in der Anzeige (von links nach rechts)
1900H	Daten für die 1. Anzeigestelle – HL-Reg.
1901H	Daten für die 2. Anzeigestelle
1902H	Daten für die 3. Anzeigestelle
1903H	Daten für die 4. Anzeigestelle
1904H	Daten für die 5. Anzeigestelle
1905H	Daten für die 6. Anzeigestelle

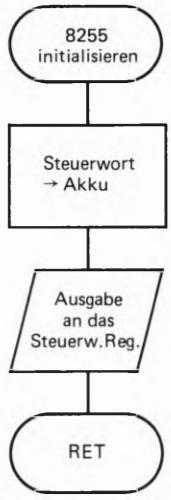


Bild S79.1
Das Unterprogramm initialisiert den Ein-/Ausgabebaustein.

Der erste Programmteil

Dieser Programmteil ist ausnahmsweise als Unterprogramm geschrieben. Das hat den einfachen Grund, daß das Programm universal einsetzbar ist. Sehen Sie sich bitte den Programmablaufplan in Bild S78.1 an.

Zu Beginn des Programms werden erst einmal alle Registerinhalte auf den Stack des Mikroprozessorsystems gerettet. Das hat den Vorteil, daß Sie sich um die Registerinhalte nicht mehr kümmern müssen, wenn das Programm von einer beliebigen Stelle aus aufgerufen wird. Sind alle Register gerettet, kann der 8255 initialisiert werden. Diesen Vorgang kennen Sie bereits, es wird einfach das Steuerwort 90H in das Steuerwortregister des Bausteins geladen. Den schematischen Programmablauf zeigt Bild S79.1.

Nun folgt ein Programmschritt, der einen Zeiger auf den ersten Speicherplatz des Anzeigepuffers setzt. Als Zeiger haben wir das HL-Registerpaar verwendet. Einfach deswegen, weil die indirekte Adressierung mit diesem Registerpaar am einfachsten ist. Die Daten aus dem Anzeigepuffer müssen ja auf möglichst einfache Weise in den Akku gebracht werden, ehe sie dann endgültig dem Port B des Ein-Ausgabebausteins zugeführt werden können. Der nächste Programmschritt definiert den Anfangsstand eines Zählers. Dieser Zähler tut weiter nichts als festzustellen, ob alle Anzeigeeinheiten schon einmal aktiviert wurden oder anders ausgedrückt: ob jede Stelle des Anzeigepuffers schon einmal angezeigt wurde.

Die nächsten Programmschritte sind als Unterprogramm geschrieben. Insgesamt sechs mal wird jeweils eine Anzeigestelle aktiviert, die entsprechenden Daten aus dem Anzeigepuffer geholt und die Segmente aktiviert. Das Zeitverzögerungs-Unterprogramm wird notwendig, weil die Einschaltzeit der einzelnen Einheiten ohne das Programm kürzer wäre als die Ausschaltzeit. Ohne das Zeitprogramm wären die zur Anzeige gebrachten Daten nur sehr schwer lesbar. Sie können das selbst ausprobieren, indem Sie diesen Programmteil einmal weglassen.

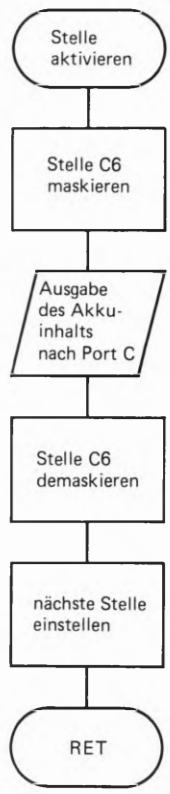


Bild S79.2
Eine Stelle der Anzeigeeinheit wird aktiviert. Das Bit Nr. 6 muß eine 1 aufweisen, weil sonst das Programm durch BREAK unterbrochen wird.

Die Unterprogramme

Das erste neue Unterprogramm trägt im Programmablaufplan den Namen „Stelle aktivieren". Als Aktivierungs- und Schieberegister dient in diesem Programmteil der Akkumulator. Er bekommt gleich zu Anfang des ersten Programms das Bitmuster 0000 0001 zugeteilt.

Daß an diesem Bitmuster noch eine Kleinigkeit fehlt, wissen Sie schon aus unserem Programm auf der Seite S 73. Die Stelle C6 des Ports C muß eine 1 aufweisen, weil sonst das $\overline{\text{BREAK}}$-Signal aktiviert wird und das Programm nicht läuft. Warum also nicht gleich an die sechste Stelle des Bitmusters eine 1 setzen? Nun, der Akkumulator soll als Schieberegister arbeiten, das bei jedem Aufruf des Unterprogramms seinen Inhalt um eine Stelle nach links schiebt. Damit wird bei jedem Aufruf eine andere Stelle der Anzeigeeinheit aktiviert.

Wäre gleich zu Beginn des Programms das Bitmuster Ø1ØØ ØØØ1 im Akku, so würde nach dem ersten Schiebevorgang daraus das Bitmuster 1ØØØ ØØ1Ø werden. Das ist aber nicht in unserem Sinne. Das richtige Bitmuster muß nach dem ersten Schiebevorgang Ø1ØØ ØØ1Ø lauten. Also macht man einen kleinen Trick. Es wird zunächst das Bitmuster ØØØØ ØØØ1 in den Akku geladen. Vor der Ausgabe an den Port C wird dieses Bitmuster aber maskiert. Das geschieht durch eine ODER-Verknüpfung des Bitmusters mit der Binärstruktur Ø1ØØ ØØØØ. Nachdem dann das gewünschte Bitmuster ausgegeben wurde, „demaskiert" man die sechste Stelle vor dem Schiebevorgang wieder.

Das Demaskieren geht am Einfachsten, wenn man das an den Port C ausgegebene Bitmuster mit Ø1ØØ ØØØØ XOR (Exklusiv-ODER) verknüpft. Nun ist die beim Schiebevorgang unerwünschte 1 an der sechsten Stelle ausgeblendet und alles läuft wie gewünscht ab. Damit ist auch schon das Unterprogramm „Stelle aktivieren" beendet.

Ein weiteres Unterprogramm heißt „Segmente aktivieren". Dieses Unterprogramm holt aus dem Anzeigepuffer jeweils ein Bitmuster und bringt dieses Bitmuster in die jeweils aktivierte Anzeige. Weiter tut dieser Programmteil nichts. Ganz einfach also. Dennoch soll auch dieser Programmteil kurz besprochen werden.

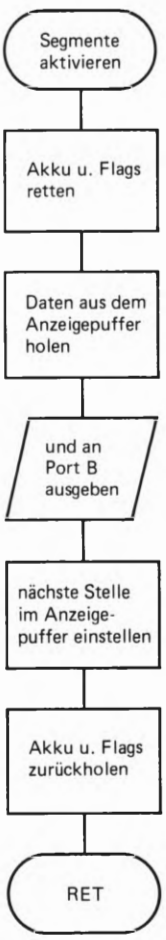


Bild S 80.1
Sowie eine Stelle der Anzeigeeinheit aktiviert ist, werden durch diesen Programmteil die einzelnen Segmente der Anzeige angesteuert.

Sehen Sie sich bitte einmal den Programmablaufplan in Bild S 80.1 an. Sie sehen daraus, daß im ersten Programmschritt der Akku auf den Steck gerettet wird. Das ist notwendig, weil wir den Akku im vorigen Unterprogramm ebenfalls benutzen. Das HL-Registerpaar dient als Zeiger, der auf den Anzeigepuffer zeigt. Wenn das Programm zum ersten Mal aufgerufen wird, hat das HL-Registerpaar den Inhalt 19ØØH. Das bedeutet, daß im zweiten Programmschritt der Inhalt der Speicherstelle 19ØØH in den Akku geholt wird und im nächsten Schritt an den Port B des 8255 weitergegeben wird. Fehlt nur noch das HL-Registerpaar für den nächsten Aufruf des Programms vorzubereiten. Das geschieht ganz einfach durch inkrementieren des HL-Registerpaars. Im Programmablaufplan heißt dieser Schritt „nächste Stelle im Anzeigepuffer einstellen". Danach muß nur noch der Akku und das Flagregister, das zu Anfang des Programms auf den Stack gerettet wurde, wieder durch einen POP-Befehl zurückgeholt werden. Damit ist auch dieses Programm beendet.

PERI 4 S80 / EDI L 69

Versuch S 80.1

Das fertige Programmlisting finden Sie auf der Seite L 69. Tasten Sie das Programm in Ihr Mikroprozessorsystem ein. Da das gesamte Programm als Unterprogramm geschrieben ist, muß zumindest ein kurzes Hauptprogramm geschrieben werden. Schreiben Sie als Hauptprogramm folgende Zeilen:

Hauptprogramm

```
1850 CD      1853 C3
1851 00      1854 50
1852 18      1855 18
```

Damit haben Sie ein kleines Hauptprogramm, das nichts weiter macht, als immer wieder das Unterprogramm Anzeige aufzurufen. Starten Sie das Programm bei der Anfangsadresse 1850H.

Beobachten Sie bitte, was geschieht, wenn Sie das Unterprogramm ZEIT nicht aufrufen. Tragen Sie zu diesem Zweck statt dem CALL-Befehl einfach drei NOP-Befehle in das Programm ein.

Aufgabe S 81.1

Im vorigen Programm sind die Inhalte der Speicherstellen 1900H bis 1905H ausschlaggebend für das, was in der Anzeige erscheinen soll. Versuchen Sie bitte die Zahlenreihe 1 2 3 4 5 6 zur Anzeige zu bringen. Zur Umcodierung verwenden Sie bitte das auf Seite H 73 gezeigte Schema.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 12.

Das Umcodierungsunterprogramm HEX 7

Sicher ist es nicht immer angenehm, wenn man eine Zahl, die zur Anzeige gebracht werden soll, erst in mühsamer Kleinarbeit in den 7-Segment-Code der Anzeige umsetzen muß. Diese Routinearbeit könnte ja durch ein Programm der Mikroprozessor selbst erledigen. Wie geht man bei der Entwicklung eines solchen Programms vor? Nun, im Grunde tut das Programm nichts anderes als das, was Sie in der vorigen Aufgabe sozusagen in Handarbeit gemacht haben.

Man macht sich zunächst eine Tabelle der codierten Zahlen. Jede Zahl wird dann einer Speicherstelle zugeordnet, genauso, als ob Sie die Tabelle irgendwo auf ein Stück Papier geschrieben hätten. Diese Tabelle haben wir schon vorgefertigt, sie sieht wie folgt aus:

Speicherstelle	Sedezimalzahl	Codierung
07F0	00	BD
07F1	01	30
07F2	02	9B
07F3	03	BA
07F4	04	36
07F5	05	AE
07F6	06	AF
07F7	07	38
07F8	08	BF
07F9	09	FE
07FA	0A	3F
07FB	0B	A7
07FC	0C	BD
07FD	0D	B3
07FE	0E	8F
07FF	0F	0F

S
82

Angenommen, die Zahl, die umcodiert werden soll, steht im Akku des Mikroprozessors. Nehmen wir weiter an, daß das HL-Registerpaar auf die erste Stelle der Umcodierungstabelle – also auf die Speicherstelle Ø7FØH – zeigt. In diesem Fall sieht der Ablauf des Umcodierungsprogramms wie folgt aus:

1. HL-Registerpaar auf den Stack retten. Das ist eine reine Vorsichtsmaßnahme, falls der Inhalt des Registerpaars nicht zerstört werden darf.

2. Das HL-Registerpaar mit der Anfangsadresse der Code-Tabelle laden. Das ist in unserem Falle die Adresse Ø7FØH.

3. Der Inhalt des Akkus darf nicht zweistellig sein. Das leuchtet ein, denn auf eine Anzeigestelle paßt nur eine einstellige Zahl. Aus diesem Grund maskiert man den Akku-Inhalt, indem der Akku-Inhalt mit der Sedezimalzahl ØFH UND-verknüpft wird.

4. Nun kommt ein wichtiger Programmschritt. Zum Inhalt des L-Registers addiert man den Akku-Inhalt. Damit zeigt das HL-Registerpaar genau auf die Stelle der Umcodierungstabelle, wo das richtige Bitmuster für die Anzeige steht. Machen Sie einen kleinen Test. Nehmen Sie an, im Akku stehe die Zahl Ø7H. Wird zum Inhalt des L-Registers diese Zahl hinzuaddiert, dann hat das L-Register den Wert F7H. Anders ausgedrückt: Die Adresse im HL-Registerpaar ist nun Ø7F7H. Wenn Sie nun in der Umcodierungstabelle nachsehen, dann steht in der Adresse Ø7F7H genau das Bitmuster, das für die Anzeige einer 7 notwendig ist.

5. Nun muß nur noch der Wert, der in der Speicherstelle Ø7F7H steht in den Akku gebracht werden. Das wiederum geht ganz einfach mit der indirekten Adressierung : LD A, (HL).

6. Nun muß nur noch der frühere Wert des HL-Registerpaars von dem Stack zurückgeholt werden. Damit ist das Unterprogramm dann auch schon zu Ende. Der Programmablaufplan in Bild S 82.1 zeigt Ihnen noch einmal die einzelnen Programmschritte dieses Programms. Sie brauchen das Programm aber nicht selbst zu schreiben, es ist im Monitorprogramm des Micro-Professors ebenso vorhanden, wie die eben beschriebene Umcodierungstabelle. Das Programm kann unter der Adresse Ø689H aufgerufen werden.

Nun haben wir uns aber lange genug mit der Anzeigeeinheit des Micro-Professors beschäftigt. Im nächsten Kapitel wenden wir uns wieder dem 8255 auf der Peripherie-Platine zu.

Bild S 82.1
Das Unterprogramm HEX 7 codiert eine Sedezimalzahl in den 7-Segment-Code. Die Zahl darf allerdings nicht größer sein als ØFH.

Setzen einzelner Bits des Ports C in Betriebsart 0

Gerade für Regelungstechnische Zwecke ist es manchmal von großer Bedeutung, daß einzelne Bits eines Ports gesetzt oder rückgesetzt werden können. Der 8255 bietet in der Betriebsart 0 die Möglichkeit, einzelne Bits des Ports C durch ein Steuerwort zu beeinflussen; wohlgemerkt, nur in der Betriebsart 0. In anderen Betriebsarten hat dieses Steuerwort eine ganz andere Bedeutung, wie Sie gleich noch sehen werden.

Der Aufbau des Steuerworts für Bit-Manipulationen am Port C

Bild S 83.1 zeigt den schematischen Aufbau des Steuerworts. Zu dem Ihnen schon bekannten Steuerwort zur Bestimmung der Funktion der einzelnen Ports, unterscheidet sich dieses Steuerwort grundsätzlich dadurch, daß das Bit Nr. 7 einen Ø-Pegel aufweist. Daran erkennt der 8255, genauer gesagt, die Steuerlogik des Bausteins, ob es sich um das eine oder andere Steuerwort handelt. Bitte merken Sie sich, daß das Steuerwort, das zur Funktionsauswahl der Ports dient, an der Stelle 7 immer eine 1 aufweist; das Steuerwort zur Beeinflussung einzelner Bits an dieser Stelle aber immer einen Ø-Pegel aufweisen muß.

Die Bits 4 bis 6 haben in diesem Steuerwort keine Bedeutung und können daher mit einer 1 oder einer Ø belegt werden.

Die Bits 1 bis 3 wählen, wie ein Binärzähler, dasjenige Bit des Ports C aus, das gesetzt oder rückgesetzt werden soll.

Das niederwertigste Bit schließlich, bestimmt, ob das angesprochene Bit gesetzt oder rückgesetzt werden soll. Eine logische Ø an der Stelle DØ bedeutet, daß das durch die Bits 1 bis 3 ausgewählte Bit am Port C eine logische Ø aufweist. Durch eine 1 an der Stelle DØ wird das ausgewählte Bit gesetzt.

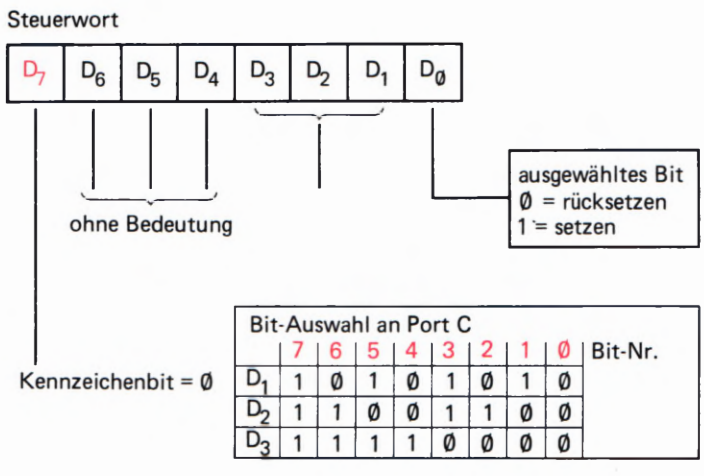


Bild S 83.1
Schema für die Bildung des Steuerworts. Mit diesem Steuerwort können einzelne Bits des Ports C gesetzt oder rückgesetzt werden. Wichtig ist, daß dieses Steuerwort an der Stelle D7 immer eine Ø aufweisen muß.

S
84

Beispiel S 84.1

Wenn Sie in der Betriebsart 0 das Bit Nr. 6 auf 1-Pegel setzen wollen, dann hat das zugehörige Steuerwort zum Beispiel folgende Bitstruktur:

Ø 1 1 1 1 1 Ø 1

Das niederwertigste Bit, also DØ ist auf 1-Pegel und deutet auf Bit-Setzfunktion. Die nächsten drei Bits sind wie ein Binärzähler ausgelegt. Die Struktur **1 1 Ø** ist eine dezimale 6. Damit wird das Bit Nr. 6 des Ports C angesprochen. Die nächsten drei Bits sind ohne Bedeutung. Wir haben diese drei Bits mit jeweils einer 1 belegt. Das höchstwertigste Bit kennzeichnet das Steuerwort und muß eine Ø aufweisen.

PERI4 S84/EDI
S84 L72

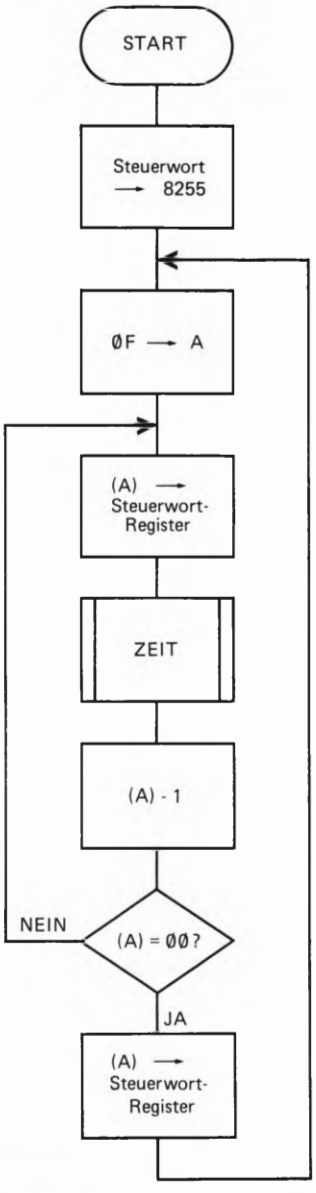


Bild S 84.1
Der Programmablaufplan zur Steuerung der Bits am Port C. Das Programm steuert jede Leuchtdiode des Ports C einmal an.

Versuch S 84.1

Nun wird es wirklich wieder einmal Zeit, etwas Praktisches zu tun. Die Auswirkung des neuen Steuerworts soll einmal ausgetestet werden. Am einfachsten können Sie diese Funktion ausprobieren, wenn Sie ein kleines Programm schreiben, das die an den Port C angeschlossenen Leuchtdioden der Reihe nach einmal aufleuchten läßt.

Was muß getan werden? Der in Bild S 84.1 dargestellte Programmablaufplan zeigt die einzelnen Programmschritte. Am Anfang des Programms wird der 8255 initialisiert. Das Steuerwort hierfür ist die Bitstruktur **1ØØØ ØØØØ.** Dieses Steuerwort besagt, daß alle Ports auf Ausgabe programmiert sind. Danach wird die Sedezimalzahl ØFH in den Akkumulator geladen, um diesen Zahlenwert im nächsten Programmschritt an das Steuerwort-Register weiterzugeben.

Die Sedezimalzahl ØFH im Steuerwort-Register bedeutet, daß das Bit Nr. 7 des Ports C eine logische 1 aufweisen soll. Bitte sehen Sie sich noch einmal das Schema auf der Seite S 83 an.

Damit Sie später den Programmablauf auch verfolgen können, wird im Programm eine kleine Zeitschleife eingebaut. Würde diese Zeitschleife nicht vorhanden sein, liefe das Programm so schnell ab, daß Sie nicht sehen könnten, was im einzelnen geschieht.

Im nächsten Programmschritt wird der Akkumulator-Inhalt dekrementiert. Das bedeutet, wenn der Akkumulator-Inhalt wieder an das Steuerwort-Register ausgegeben wird, ist das Bit Nr. 7 des Port C zurückgesetzt. Nach jedem Dekrementieren fragt das Programm, ob der Inhalt des Akkumulators schon den Wert ØØ erreicht hat. Ist dies nicht der Fall, springt das Programm an diejenige Stelle zurück, an der der Akkumulator-Inhalt an das Steuerwort-Register weitergegeben wird. Im anderen Fall, wenn der Akkumulator schon den Wert ØØ angenommen hat, wird zu der Stelle im Programm verzweigt, wo der Akku mit ØFH geladen wird. Das Spiel kann erneut beginnen.

Bitte überprüfen Sie selbst die Bedeutung der einzelnen Steuerwörter, die durch den Zählvorgang im Akkumulator eingestellt werden. Nehmen Sie dazu das Schema auf Seite S 83 zu Hilfe. Das komplette Programm finden Sie auf der Seite L 72.

Tasten Sie das Programm bitte in Ihr Mikroprozessor-System ein. Denken Sie bitte daran, daß nur die obere Flachbandleitung vom Micro-Professor zur Peripherie-Platine gesteckt sein darf. Die steckbaren Brücken, die über Ein- oder Ausgabefunktion auf der Platine entscheiden, brauchen nicht gesteckt zu sein. Wenn Sie das Programm starten, sehen Sie, wie die einzelnen Leuchtdioden der Reihe nach von links nach rechts angesteuert werden.

Bedeutung des Steuerworts in der Betriebsart 1

Wie Sie gesehen haben, kann durch ein Steuerwort, das durch das Bit Nr. 7 gekennzeichnet ist, der Port C beeinflußt werden; dies geschieht bitweise. Wir haben auch schon ausdrücklich darauf hingewiesen, daß die Beeinflussung der einzelnen Bits durch dieses Steuerwort nur in der Betriebsart 0 so, wie besprochen funktioniert.

Sowie der 8255 in einer höheren Betriebsart arbeitet, hat dieses Steuerwort eine ganz andere Bedeutung. Diese Bedeutung wollen wir nun etwas näher betrachten.

Wie Sie sich bestimmt erinnern, haben wir im Fachgebiet Hardware die Bedeutung der einzelnen Steuersignale besprochen. Weiter haben wir gesagt, daß ein Interrupt-Signal nur dann zum Mikroprozessor gelangen kann, wenn das entsprechende interne INTE-Flipflop gesetzt ist. Lesen Sie sich bitte noch einmal kurz den auf der Seite H77 beschriebenen Vorgang bei der Interrupt-Verarbeitung durch.

Jedem Port ist demnach ein INTE-Flipflop zugeordnet. Das bedeutet, daß diese Flipflops durch das Programm irgendwie gesteuert werden müssen. Diese Steuerung erfolgt durch das eben besprochene Steuerwort. Am Steuerwort selbst brauchen Sie gar nichts zu ändern. Dadurch, daß der 8255 in der Betriebsart 1 oder 2 arbeitet, weiß der Baustein, daß durch das Steuerwort nicht die Bits des Ports C angesprochen werden, sondern eines der internen INTE-Flipflops.

Sperren oder Freigeben der INTE-Flipflops in der Betriebsart 1

Sehen Sie sich bitte nochmals den Aufbau des Steuerworts für die Bitmanipulation auf der Seite S 83 an. Daraus ersehen Sie, daß das niederwertigste Bit des Steuerworts darüber entscheidet, ob ein Bit gesetzt oder rückgesetzt werden soll. Genauso, wie das in der Betriebsart 0 für die Bits des Ports C gilt, genauso gilt diese Aussage für das Setzen oder Rücksetzen der INTE-Flipflops in der Betriebsart 1. Zusammenfassend gilt:

Der 8255 arbeitet in der

Betriebsart 0	Betriebsart 1 oder 2
Die einzelnen Bits des Ports C werden angesprochen	Eines der internen INTE-Flipflops wird angesprochen

Wie weiß man, mit welchem Bit welches INTE-Flipflop angesprochen wird? Das ist in der Tat nicht ohne weitere Kenntnisse eines Datenblattes des 8255 möglich. Eine kleine „Eselsbrücke" gibt es aber auch für diesen Fall.

S
86

Ist ein Port auf Eingabe programmiert, dann gilt:

Das zuständige Bit für das INTE-Flipflop hat dieselbe Nummer wie die Bitnummer des $\overline{\text{STB}}$-Signals. Wenn zum Beispiel der Port A auf Eingabe programmiert ist, dann (vgl. Bild H76.1) ist das Bit Nr. 4 des Ports C zuständig für das $\overline{\text{STB}}$-Signal. Das heißt, daß für das Sperren oder Freigeben des INTE-Flipflops das Bit mit der Nummer 4 zuständig ist.

Ist ein Port auf Ausgabe programmiert, dann gilt:

Das zuständige Bit für das INTE-Flipflop hat dieselbe Nummer wie die Bitnummer des $\overline{\text{ACK}}$-Signals. Auch hierzu ein Beispiel: Ist der Port B auf Eingabe programmiert, dann erscheint das $\overline{\text{ACK}}$-Signal an Bit Nr. 2 des Ports C. Das heißt, daß auch das zuständige INTE-Flipflop durch das Bit Nr. 2 beeinflußbar ist.

Wir haben die Möglichkeiten der Steuerwortbildung in dem folgenden Schema zusammengefaßt. Aus dieser Zusammenfassung können Sie erkennen, wie die Bitstruktur der einzelnen Steuerwörter für die Beeinflussung der INTE-Flipflops aussehen. Die Zusammenfassung gilt für die Betriebsart 1. Das mit einem X gekennzeichnete Bit in der Bitstruktur des Steuerworts sperrt durch eine Ø an dieser Stelle das Interrupt-Signal; durch eine 1 wird das entsprechende INTE-Flipflop gesetzt – also das Interrupt-Signal freigegeben.

Betriebsart 1				Steuerwort für	
PORT A	PORT B	INTE A	INTE B	INTE A	INTE B
Eing.	Eing.	C4	C2	Ø111 100X	Ø111 010X
Ausg.	Eing.	C6	C2	Ø111 110X	Ø111 010X
Eing.	Ausg.	C4	C2	Ø111 100X	Ø111 010X
Ausg.	Ausg.	C6	C2	Ø111 110X	Ø111 010X

1 an dieser Stelle bedeutet Interrupt-Freigabe
Ø bedeutet Interrupt gesperrt

Sie haben nun wieder eine ganze Menge Theorie gelesen, und es ist angebracht, durch kleine Beispiele diese zunächst bestimmt etwas umständliche Arbeitsweise besser in den Griff zu bekommen.

Beispiel S 86.1

Der Ein- Ausgabebaustein 8255 soll so vorprogrammiert werden, daß beide Ports in der Betriebsart 1 arbeiten. Port A soll auf Ausgabe pro-

grammiert werden; Port B soll dagegen als Eingabeport arbeiten. Diejenigen Bits, die an Port C nicht zur Bedienung der Steuersignale benötigt werden, sollen auf Ausgabefunktion programmiert werden. Welche Programmschritte müssen getan werden, wenn der Baustein initialisiert werden soll und beide INTE-Flipflops freigegeben werden sollen?

Zuerst muß das geeignete Steuerwort für die Initialisierung gefunden werden. Dazu können Sie sich die Tafel T 6 oder das Schema auf der Seite H 69 zu Hilfe nehmen. Wir haben die Bitstruktur schon zusammengetragen, es ist dies: 1 Ø 1 Ø Ø 1 1 1.

Versuchen Sie aber einmal selbst das Steuerwort zu finden. Übrigens, das niederwertigste Bit in der angegebenen Struktur entscheidet laut Schema über Ein- oder Ausgabefunktion der niederwertigen Hälfte des Ports C. Diese niederwertige Hälfte ist aber, wenn beide Ports in der Betriebsart 1 arbeiten, immer voll mit Steuersignalen belegt. Es spielt also keine Rolle, ob Sie Bit DØ des Steuerworts mit einer 1 oder einer Ø belegen – eine Stolperfalle, über die schon so mancher Praktiker bei der Festlegung des Steuerworts gefallen ist! Wir haben diese Stelle des Steuerworts mit einer 1 belegt; so wie wir es eigentlich immer tun, wenn es egal ist, ob eine Stelle eine Ø oder eine 1 aufweist.

Mit der Ausgabe des ausgewählten Bitmusters an das Steuerwort-Register ist der 8255 initialisiert. Nun gilt es, die internen INTE-Flipflops zu setzen. Die Bitstruktur der notwendigen Steuerwörter finden Sie im Schema auf der Seite S 86. Daraus erkennen Sie die Bitstruktur Ø 1 1 1 1 1 Ø 1 für INTE A, und die Bitstruktur Ø 1 1 1 Ø 1 Ø 1 für INTE B.

Diese beiden Steuerwörter können nun nacheinander an das Steuerwort-Register durch einen OUT-Befehl weitergegeben werden. Beim ersten Hinsehen könnte man sagen: „Schon eine Viecherei, diesen Baustein zu programmieren". Wenn Sie den Vorgang aber ein paar mal gemacht haben, sieht alles nicht mehr so schlimm aus. Deshalb gehen wir gleich einen Schritt weiter und schreiben auch noch die Befehlsfolge für diese Programmschritte; wie Sie es nun schon gewohnt sind, im Assembler-Format.

```
;*****************************************************************
;Initialisierungsprogramm
;
;START LD      A,ØA7H      ;Steuerwort in den Akku laden
      OUT     (ØC3H),A    ;Ausgabe an das Steuerwort-
                          ;Register.
      LD      A,Ø7DH      ;Steuerwort zur Freigabe von
                          ;INTE A in den Akku laden und
      OUT     (ØC3H),A    ;in das Steuerwort-Register
                          ;geben.
      LD      A,Ø75H      ;Steuerwort zur Freigabe von
                          ;INTE B in den Akku laden
      OUT     (ØC3H),A    ;und in das Steuerwort-Register
                          ;geben.
```

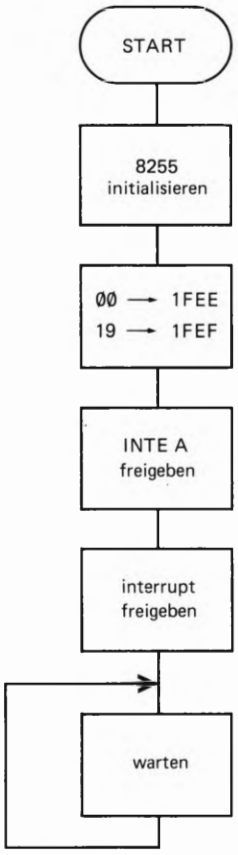


Bild S 88.1
Das Hauptprogramm arbeitet fast nur als Initialisierungsprogramm und als Warteschleife. In einem größeren Programm wird das Hauptprogramm in der Regel andere wichtigere Tätigkeiten ausführen. Wir haben aber das Hauptprogramm so kurz wie möglich geschrieben, damit es übersichtlich ist.

Programmieren in der Betriebsart 1 – Dateneingabe

Sie haben im vorigen Fachgebiet Hardware dieses Lehrgangs schon einiges über den Ein- Ausgabebaustein 8255 gelesen, wenn dieser in der Betriebsart 1 arbeiten soll. Sie wissen nun bereits, daß in der Betriebsart 1 Steuersignale zur Ein- oder Ausgabe von Daten an Port C anstehen. Weiter ist Ihnen auch schon bekannt, daß in der Betriebsart 1 hauptsächlich mit Interrupts gearbeitet wird, denn erst dadurch können manche Probleme mit einem Mikroprozessor gelöst werden, die sonst nur sehr umständlich zu programmieren wären.

Von der Behandlung der PIO und des CTC-Bausteins her, ist Ihnen bestimmt auch schon bekannt, daß die Auswirkungen auf einen Interrupt ganz verschieden sein können; je nachdem, wie die PIO programmiert ist. Wenn Sie mit der Peripherie-Platine und dem 8255 arbeiten, bietet das System nur eine Interrupt-Möglichkeit, den sogenannten RESTART 38. Sie haben mit diesem Interrupt und dem RESTART 38 schon auf der Seite S 75 Bekanntschaft gemacht. Bevor wir nun die Betriebsart 1 per Programm ausprobieren, lesen Sie sich bitte den Abschnitt Interrupt mit RESTART RST 38 durch, denn zum Verständnis des folgendes Programms ist es wichtig, daß Sie die Vorgangsweise bei der Interrupt-Verarbeitung noch in Erinnerung haben.

Beispiel S 88.1

In der Betriebsart 1 soll eine einfache Dateneingabe gemacht werden. Der 8255 wird so programmiert, daß Port A als Eingabeport arbeitet. Port B ist auf der Peripherie-Platine nicht zugänglich. Es spielt deshalb keine Rolle, welche Datenflußrichtung für diesen Port festgelegt wird. Der Einfachheit halber programmieren wir ihn ebenfalls auf Dateneingabe.

Das Hauptprogramm selbst soll nichts weiter tun, als den 8255 zu initialisieren und auf einen Interrupt zu warten. Das ist zwar eine recht unsinnige Arbeit, zum Testen der Betriebsart reicht das aber vollkommen aus.

Durch die Interrupt-Service-Routine, die mit der Adresse 19ØØH beginnen soll, wird das an Port A anstehende Byte in das B-Register eingelesen. Damit wir auch wissen, daß er das getan hat, soll uns der Baustein ein Zeichen geben. Sie erinnern sich, wenn Port A und Port B in der Betriebsart 1 arbeiten und auf Eingabe programmiert sind, sind immerhin noch zwei Bits des Ports C frei, die nicht für Steuersignale benutzt werden. Diese beiden Bits benutzen wir sozusagen als Signal. Wenn die Daten von Port A eingelesen sind, soll das Bit Nr. 7 mit einem 1-Signal eine Sekunde lang belegt werden.

Die obere Leuchtdiodenreihe auf der Peripherie-Platine ist ja mit dem Port C verbunden. Beim Programmablauf brauchen Sie also nur die Leuchtdiode 7 zu beobachten, um festzustellen, ob das Programm auch wunschgemäß arbeitet. Voraussetzung ist natürlich, daß bei der Initialisierung des Bausteins dieses Bit des Ports C auch als Ausgabebit berücksichtigt wird. Der Programmablaufplan in Bild S 88.1 und S 89.1 zeigt noch einmal deutlich, was das eben besprochene Programm leisten soll.

Der Programmablauf

Das Bild S 88.1 zeigt den Programmablaufplan des Hauptprogramms. Dieses Hauptprogramm tut im Grunde nicht allzuviel, außer den Ein-Ausgabebaustein zu initialisieren. Zur Initialisierung muß zunächst das geeignete Steuerwort gefunden werden. Bevor Sie im fertigen Listing auf der Seite L 73 nachsehen, sollten Sie versuchen, dieses Steuerwort selbst zu finden.

A, B Ein
Rest C aus
Mode 1

Aufgabe S 89.1

Ermitteln Sie das Steuerwort für die Betriebsart 1 des 8255, wenn Port A und Port B als Eingabeports arbeiten und die an Port C freibleibenden Bits C 6 und C 7 als Ausgabebits arbeiten sollen. Tragen Sie die Bitstruktur in das vorgefertigte Schema ein.

D7	D6	D5	D4	D3	D2	D1	DØ	gesuchtes Steuerwort
1	0	1	1	Ø	1	1	1	bitte hier eintragen.

B 7H

Die Lösung der Aufgabe finden Sie auf der Seite Ü 12.

Der zweite Programmschritt im Hauptprogramm legt die Anfangsadresse der Interrupt-Service-Routine fest. Sie erinnern sich, wenn einem Interrupt mit nachfolgendem RST 38 stattgegeben wird, sucht sich das Monitorprogramm des Micro-Professors die Anfangsadresse der Interrupt-Service-Routine in den Adressen 1FEEH und 1FEFH.

Mode Ø

Im folgenden Programmschritt wird das INTE A-Flipflop des 8255 gesetzt. Das geschieht wieder mit dem Steuerwort für Bit setzen/rücksetzen. Außerdem muß im nächsten Programmschritt auch der Mikroprozessor für Interrupts empfänglich gemacht werden. Dies erreicht man durch den Befehl EI. Nun sind alle Vorbereitungen getroffen; das Hauptprogramm kann in einer Warteschleife verweilen, bis der erste Interrupt eintrifft.

Das Bild S 89.1 zeigt den Programmablaufplan für die Interrupt-Service- Routine. Hier geschieht nichts Weltbewegendes, denn es wird zu Beginn des Programms lediglich ein erneut eintreffender Interrupt gesperrt und anschließend werden die an Port A anstehenden Daten in den Akku eingelesen. Vom Akku gelangen sie dann anschließend in das B-Register.

Erwähnenswert ist aber in diesem Programmablauf doch noch das Setzen des Bits C 7. Wenn man sich ein Datenblatt des 8255 vornimmt, könnte man meinen, die nicht mit Steuersignalen belegten Bits des Ports C können einfach durch einen OUT-Befehl an Port C geliefert werden. Sie können dies gerne einmal ausprobieren, aber es sei hier schon gesagt, daß das nicht gehen wird. Ganz im Gegenteil, man muß

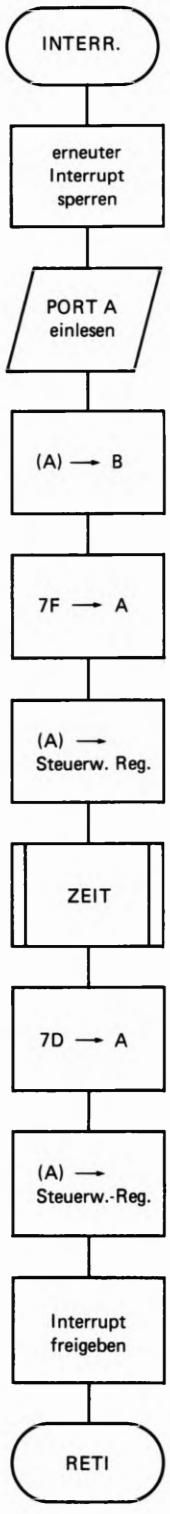
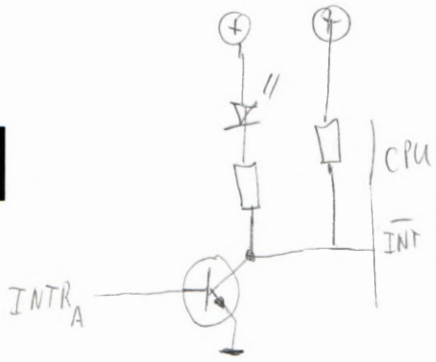


Bild S 89.1
Die Interrupt-Service-Routine bedient hauptsächlich das Bit Nr. 7 des Ports C. Sowie ein Interrupt-Signal eingetroffen ist, wird dies eine Sekunde lang an Bit C 7 angezeigt.

S
90

über das Steuerwort für die Bitmanipulation das gewünschte Bit setzen oder rücksetzen. Nach unserer Aufgabenstellung soll das Bit C7 gesetzt werden. Das bedeutet, daß die Bitstruktur für dieses Steuerwort wie folgt aussehen muß: Ø 1 1 1 1 1 1 1. Bitte überprüfen Sie diese Bitstruktur nochmals selbst, indem Sie sich das Schema auf der Seite S 83 zu Hilfe nehmen. Versuchen Sie auch, die Bitstruktur für das Rücksetzen dieses Bits zusammenzustellen. Laut Aufgabenstellung soll dieses Bit nur ca. eine Sekunde lang auf 1-Pegel bleiben und dann wieder Ø-Pegel annehmen. Dies geht auch aus dem Programmablaufplan hervor.



Dem Unterprogramm ZEIT haben wir keinen Programmablaufplan gewidmet. Dieses Unterprogramm haben Sie innerhalb der letzten Programme so oft verwendet, daß es Ihnen nun schon zu genüge bekannt ist. Das vollständige Programmlisting finden Sie auf der Seite L 73.

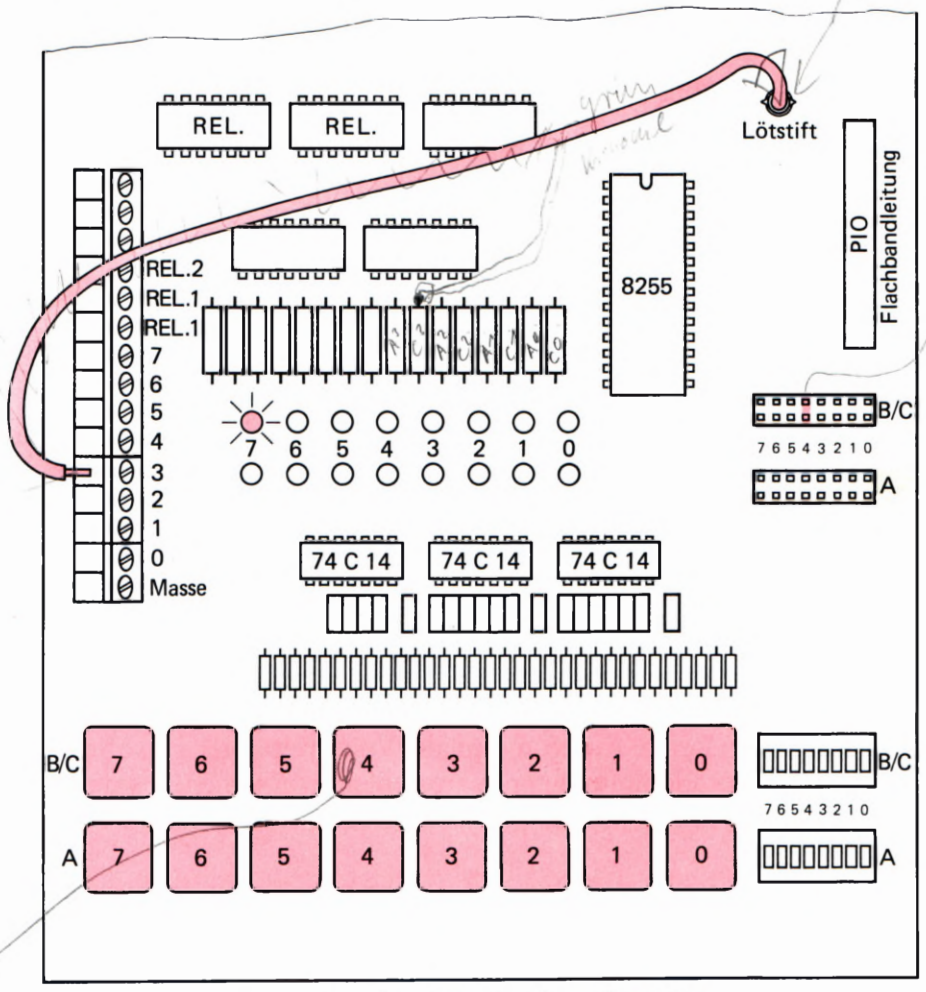


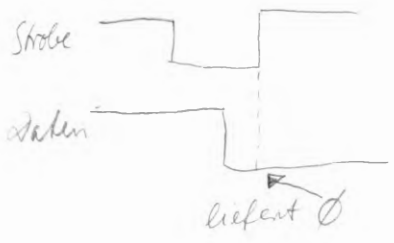
Bild S 90.1
Die in rot eingezeichneten Verbindungen sind für die Durchführung des Versuchs wichtig. Die Peripherie-Leiterplatte ist hier in vereinfachter Form dargestellt.

Versuchsdurchführung

Wenn Sie das Programm ausprobieren wollen, muß zu allererst die Peripherie-Platine für das Programm vorbereitet werden. Was im einzelnen zu machen ist, ersehen Sie aus dem Bild S 90.1. Wichtig ist, daß Sie die Verbindung von der Anschlußklemmleiste zum Lötstift herstellen. Die mit der 3 gekennzeichnete Anschlußklemme führt in der

eben besprochenen Programmierung des 8255 das Interrupt-Signal. Über den Lötstift wird das Interrupt-Signal dann an den Mikroprozessor weitergeführt. Die rot eingezeichnete steckbare Brücke stellt die Verbindung der Taste zum Bit C 4 des Ports C her. Dieses Signal ist das STB-Signal, mit dem Sie während des Programmablaufs dem 8255 bekanntgeben können, daß Daten zum Einlesen bereitstehen.

Vergessen Sie auch nicht, die Platine mit Spannung zu versorgen und die Flachbandleitung anzuschließen. Es darf nur die Flachbandleitung für den 8255 angeschlossen werden. Die Flachbandleitung zur PIO muß gelöst sein. So, nun sind alle Vorbereitungen getroffen.



Nun können Sie das Programm in das Micro-Professor-System eintasten. Das Programm finden Sie auf der Seite L 73. Wenn das Programm eingetastet ist, dann starten Sie es bitte mit der Anfangsadresse 18ØØH. Die Anzeige des Micro-Professors wird dann dunkel. Betätigen Sie die Taste 4 in der oberen Tastenreihe (im Bild S 90.1 rot gekennzeichnet). Bei jeder Betätigung muß die Leuchtdiode Nr. 7 in der oberen Leuchtdiodenreihe für ca. eine Sekunde aufleuchten und dann wieder dunkel bleiben. Dies ist ein Zeichen dafür, daß ein Interrupt stattgefunden hat.

PERIS91/EDI:3
S 91 L75

Versuch S 91.1

Um zu zeigen, daß es auch andere Anwendungsmöglichkeiten für das Arbeiten in der Betriebsart 1 gibt, finden Sie auf der Seite L 75 ein weiteres Programm. Im letzten Programm haben wir zwar die Daten, die an Port C anstehen, in den Akkumulator eingelesen, haben aber keinen Wert darauf gelegt, die Daten auch anzuzeigen. Wichtiger war es, zunächst ein einfaches Programm zu haben, das auf möglichst einfache Weise zeigt, daß auch tatsächlich ein Interrupt stattfindet, wenn das STB-Signal an Port C betätigt wird.

Im Programm auf der Seite L 75 wird im Grunde das gleiche gemacht. Es werden Daten, die am Port A anstehen, in den Akku eingelesen. Der Unterschied zum vorigen Programm ist allerdings, daß diese Daten dann über ein im Micro-Professor-Monitorprogramm befindliches Anzeigeprogramm auf der Anzeige sichtbar gemacht werden.

Nach jedem Interrupt, also wenn Sie die Taste C 4 betätigen, wird die an Port A eingelesene Zahl auf der 7-Segment-Anzeige des Micro-Professors angezeigt. Allerdings darf diese Zahl nicht größer sein, als ØFH, weil nur eine 7-Segment-Anzeige zur Darstellung der Zahl benutzt wird.

Wenn Sie die Hardware noch so präpariert aufgebaut haben, wie es das letzte Programmbeispiel gefordert hat, sollten Sie das Programm auf der Seite L 75 einmal eintasten und ausprobieren. Vielleicht können Sie doch einige Anregungen für eigene Anwendungsmöglichkeiten aus diesem Programmbeispiel entnehmen. Doch nun gleich einen Schritt weiter; zur Datenausgabe.

S
92

Programmieren in der Betriebsart 1 – Datenausgabe

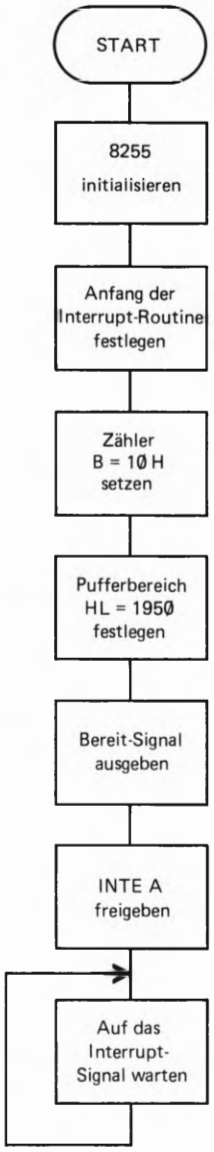


Bild S 92.1
Der Programmablaufplan für das Hauptprogramm. Das Programm leistet die Initialisierungsarbeit für den 8255. Ansonsten wartet das Programm auf einen eintreffenden Interrupt.

Wie Daten in der Betriebsart 1 vom Mikroprozessor zu einem der Ports des 8255 gelangen und welche Bedeutung die einzelnen Steuersignale dabei haben, ist das Thema des folgenden Abschnitts dieses Lehrbriefs. Im Fachgebiet Hardware haben Sie schon erfahren, wie der prinzipielle Ablauf bei der Datenausgabe erfolgt. Es gilt nun, diese Kenntnisse auch in der Praxis auszuprobieren.

Fassen wir noch einmal kurz zusammen, was bei der Datenausgabe zu beachten ist:

1. Das richtige Steuerwort für die Datenausgabe in der Betriebsart 1 muß gefunden werden.
2. Über das Steuerwort für die Bitmanipulation muß das entsprechende INTE-Flipflop gesetzt werden, damit das Interrupt-Signal freigegeben ist.
3. Die Steuersignale am Port C des 8255 sind zu bedienen. Es ist dies einmal das $\overline{\text{OBF}}$-Signal, das der 8255 ausgibt, sobald er Daten vom Mikroprozessor erhalten hat. Sobald das $\overline{\text{OBF}}$-Signal aktiv ist, können die Daten von der Peripherie abgeholt werden. Daß dies geschehen ist, gibt die Peripherie durch ein Ø-Signal am $\overline{\text{ACK}}$-Eingang des 8255 bekannt. Dadurch kann ein Interrupt stattfinden, der den Mikroprozessor veranlaßt, erneut Daten zu liefern.

Versuch S 92.1

Im nun folgenden Programm können Sie die Arbeitsweise bei der Datenausgabe in der Betriebsart 1 verfolgen. Das Programm ist wieder so kurz wie möglich gehalten, damit es gut überschaubar ist. Was soll nun das Programm tun?

Ganz einfach, wir definieren einen bestimmten Speicherbereich. Die Daten, die sich in diesem Speicherbereich befinden, werden Schritt für Schritt an den 8255 geliefert; und zwar an Port A. Welche Speicherzellen als auszulesender Bereich definiert werden, ist im Grunde egal. Wichtig ist nur, daß der Speicherbereich außerhalb des Programmbereichs liegt. Das ist aber kein Muß, sondern ein Kann. Wir kommen gleich noch einmal darauf zu sprechen. Nehmen wir es einfach hin, der auszulesende Speicherbereich, den wir kurz Puffer nennen wollen, soll mit der Adresse 1950H beginnen. Insgesamt sollen 15 Speicherinhalte übertragen werden. Soweit die Grundforderungen. Doch nun zum Programm selbst.

Programmbeschreibung

In Bild S 92.1 ist der Programmablaufplan des Hauptprogramms abgebildet. Das Programm beginnt damit, daß der 8255 initialisiert wird. Nehmen wir an, daß beide Ports – also Port A und der für uns auf der Platine unzugängliche Port B – auf Datenausgabe programmiert werden. Ebenfalls die übrigbleibenden Bits des Ports C, die nicht für Steuersignale benutzt werden, sollen auf Datenausgabe programmiert sein.

Mit diesen Forderungen entsteht für das Steuerwort die Bitstruktur **1 0 1 0 0 1 0 1.** Sedezimal ausgedrückt, ist dies die Zahl A5H. Diese Zahl wird in das Steuerwortregister des 8255 eingeschrieben, womit der Baustein dann auch schon initialisiert ist.

Der nächste Programmschritt ist Ihnen bestimmt nun auch schon geläufig. Die Adresse der Interrupt-Service-Routine muß festgelegt werden. Wo diese Routine beginnt, ist eigentlich egal; sie muß nur irgendwann festgelegt werden. Wir haben den Beginn der Routine bei der Adresse 1900H festgelegt. Diese Adresse wird sozusagen in 1FEEH und 1FEFH hinterlegt. Aber dieses Spiel kennen Sie schon.

Nun folgt die Festlegung der Register. Zum einen haben wir das B-Register als Zähler definiert, zum anderen arbeitet das HL-Registerpaar als Zeiger, der auf den Anfangsbereich des Puffers zeigt.

Damit der Anwender des Programms auch merkt, daß das Programm die Vorarbeit für eine Interrupt-Verarbeitung geleistet hat, geben wir das am Port C bekannt. Die Leuchtdiode C4 soll aufleuchten, sobald das Programm bereit ist, einen Interrupt zu verarbeiten. Die Leuchtdiode wird im Programm durch die Bitmanipulation am Port C aktiviert. Wie die Bitstruktur für das Steuerwort gebildet wird, erfahren Sie aus dem Schema, das auf der Seite S 89 dargestellt ist. Nehmen wir es ruhig vorweg, es ist dies die Sedezimalzahl 79H. Im Programmablaufplan haben wir diesen Programmschritt „Bereit-Signal ausgeben" genannt.

Der letzte Programmpunkt des Hauptprogramms ist nun gekommen; es muß noch das INTE A-Flipflop gesetzt werden. Aber auch dies ist keine Schwierigkeit, wenn Sie nach dem Schema auf der Seite S 83 das geeignete Steuerwort zusammenstellen und wissen, daß das Bit Nr. 6 gesetzt werden muß. Wenn alle Stricke reißen, dann nehmen Sie sich einfach die Tabelle auf der Seite S 86 vor, dort finden Sie eine Zusammenstellung, wie das Steuerwort aussehen muß, wenn der Port A als Ausgabeport in Betriebsart 1 arbeitet und das INTE-Flipflop gesetzt werden soll.

Aufgabe S 93.1

Versuchen Sie bitte das Steuerwort herauszusuchen, das notwendig ist, um das INTE-Flipflop zu setzen, wenn der Port A des 8255 in der Betriebsart 1 als Ausgabeport arbeitet.

Die Lösung dieser Aufgabe finden Sie auf der Seite Ü 12.

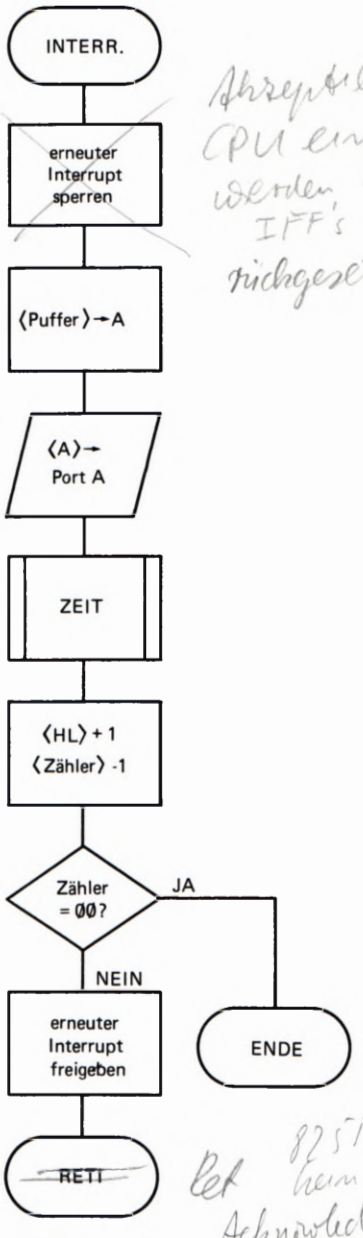


Bild S 93.1
Die Interrupt-Service-Routine. Sie wird aufgerufen, wenn vom 8255 ein Interrupt-Signal an den Mikroprozessor gelangt.

Die Interrupt-Service-Routine

Das Bild S 93.1 zeigt den Programmablaufplan der Interrupt-Service-Routine. Dieser Programmteil beginnt mit der Adresse 1900H. Zu Beginn des Programms wird ein erneut eintreffendes Interruptsignal gesperrt. Danach folgt das Abrufen einer Speicherzelle aus dem Pufferbereich. Der Inhalt der Speicherzelle wird danach im nächsten Programmschritt an den Port A weitergegeben und zwar durch einen einfachen OUT-Befehl.

S

94

Damit Sie den Programmablauf an der Reaktion der Leuchtdioden auf der Peripherie-Platine auch verfolgen können, wird im nächsten Programmschritt eine Zeitschleife aufgerufen, die das Programm in eine ca. 1 Sekunde dauernde Warteschleife bringt.

Ist die Zeitschleife durchlaufen, stellt das Programm die Adresse der nächsten Speicherstelle ein, deren Inhalt beim Eintreffen des nächsten Interrupt-Signals wieder an den Port A weitergegeben wird. Ebenfalls muß der Zähler, der die Anzahl der aufzurufenden Speicherstellen festlegt, dekrementiert werden. Ist der Zählerstand Null, dann sind alle Speicherstellen des Puffers einmal aufgerufen worden und das Programm ist beendet. Im anderen Fall, wenn der Zählerstand noch nicht Null ist, gibt das Programm einem erneuten Interrupt statt und kehrt in das Hauptprogramm zurück.

PER14 S94/EDI:3

S94 L77

Versuchsdurchführung

Präparieren Sie Ihre Peripherie-Leiterplatte so, wie es im letzten Versuch schon beschrieben wurde. Alle Verbindungen bleiben gleich. Lediglich die steckbaren Brücken müssen anders gesteckt werden. Das ist auch leicht einzusehen, denn die Ports arbeiten nun als Datenausgabeports. Im vorigen Versuch waren es ja Dateneingaben. Deswegen ändern sich auch die Steuersignale.

Entfernen Sie alle steckbaren Brücken. Nur eine Brücke muß noch geschlossen sein, und zwar die **Brücke C6.** Damit bedienen Sie gleich das $\overline{\text{ACK}}$-Signal, denn Sie müssen während des Programms durch Tastendruck „Peripherie spielen".

Wenn die Peripherie-Platine wie eben besprochen präpariert ist, können Sie das Programm (Seite L 77) eintasten und bei der Adresse 1800H starten. Die 7-Segment-Anzeigen werden dunkel. Achten Sie darauf, daß Sie gleich nach dem Start des Programms die Taste Nr. 6 bedienen. Sowie Sie die Taste betätigen, wird an Port A der Inhalt einer neuen Speicherzelle sichtbar. Wenn Sie ganz sicher sein wollen, daß das Programm richtig arbeitet, können Sie den Adreßpuffer mit markanten Bitstrukturen direkt über die Tastatur des Micro-Professors laden.

Die Arbeitsweise des Programms ist im Grunde nichts anderes als das, was bei der Übertragung eines Speicherbereichs in ein anderes Mikroprozessor-System oder zu einem Drucker getan wird. Somit kann natürlich auch die Aufgabe entstehen, daß sich das Programm selbst übertragen soll.

1800 C: 7 6 5 4 3 2 1 0

↓ ↑ ↓ ↓

OBF AckA IBF INTRA

(ANZ: 2)

Port A S94/EDI:3 mit more Time

Bit 5 4

0 1 Bereit

1 0 im ISR. Wait

1 1 " " nach Out "

Prüfungsaufgaben

Prüfungsaufgaben

Was ist beim Anfertigen und Einsenden der Lösungen der Prüfungsaufgaben zu beachten?

Bitte senden Sie die Lösungen der Prüfungsaufgaben nur dann an das Lehrinstitut ein, wenn Sie diesen Lehrgang als Fernunterrichtswerk erworben haben. Wenn Ihnen der Lehrgang als Selbstunterrichtswerk zur Verfügung steht, dann können Sie die Prüfungsaufgaben an das Lehrinstitut zur Korrektur einsenden, wenn Sie zusätzlich ein entsprechendes Betreuungspaket erwerben.

Zur Einsendung sind nur die Lösungen der Prüfungsaufgaben bestimmt, die Sie auf den Seiten mit der Griffmarke P – in diesem Lehrbrief ab der Seite P 3 – vorfinden. Die in den Abschnitten gestellten Aufgaben, zu denen wir Ihnen jeweils im selben Lehrbrief auf den Ü-Seiten die Lösungen angeben, dienen nur Ihrer Selbstkontrolle. Ihre Lösungen dieser Übungsaufgaben sollen Sie nicht an uns einschicken.

Beschreiben Sie bei Ihren Aufgaben-Lösungen bitte grundsätzlich jedes Blatt nur auf einer Seite! Der Papierverbrauch wird dadurch zwar größer, aber Sie erleichtern damit dem Korrektor die Arbeit.

Ihre Studiennummer und Ihr Name sollen auf jedes Blatt geschrieben werden, damit es keine Verzögerungen oder Falschsendungen gibt. Ihre Anschrift braucht nur auf dem ersten Lösungsblatt zu stehen.

Einen Gutschein für weitere Aufgabenlösungsblätter erhalten Sie mit den ersten Lösungsblättern. Füllen Sie den Gutschein bitte aus und senden Sie ihn mit den ersten Aufgabenlösungen an uns ein. Sie erhalten die bestellten Blätter kostenlos und portofrei. Der Sendung liegt wieder ein Gutschein für Ihre nächste Bestellung bei, so daß Sie keine Lösungsblätter hinzukaufen müssen.

Ferner legen wir Briefumschläge verschiedener Größe bei. Verwenden Sie zum Einsenden Ihrer Prüfungsarbeiten an uns einen dieser bereits mit unserer Anschrift versehenen Briefumschläge.

Die großen Umschläge werden auf dem Postweg leicht beschädigt, wenn sich nur wenige Blätter darin befinden. Wenige Blätter falten Sie besser und stecken sie in einen kleinen Umschlag.

Legen Sie Ihrer Sendung bitte einen gleichgroßen Umschlag bei und vergessen Sie nicht, auf diesen Ihre Anschrift zu schreiben. In diesem Umschlag erhalten Sie Ihre Arbeiten zurück. Wenn Sie mit Ihren Arbeiten einen Gutschein zum Bezug von Lösungsblättern einsenden, dann legen Sie bitte auf jeden Fall den mit Ihrer Anschrift versehenen großen Umschlag bei.

Die für Ihre erste Korrektursendung nicht benötigten Briefumschläge bewahren Sie bitte auf, falls Sie zum Einsenden der Aufgabenlösungen späterer Lehrbriefe andersformatige Umschläge benötigen.

Wir legen Ihnen bei jeder Rücksendung wieder einen gleichgroßen, an uns adressierten Umschlag und einen weiteren bei, auf den Sie bitte wieder Ihre Anschrift schreiben. Auf diese Weise haben Sie bei jeder Sendung an uns die Möglichkeit, das Umschlagformat der jeweiligen Anzahl der Blätter anzupassen.

Frankieren Sie bitte den an uns gerichteten Brief. Das Porto für die Rücksendung tragen wir. Sie brauchen also den Umschlag, den Sie – mit Ihrer Adresse versehen – Ihrer Sendung an uns beilegen, nicht zu frankieren.

Sie können die Lösungen mehrerer Lehrbriefe zusammen an uns einsenden. Es ist jedoch zweckmäßig, wenn wir Sie recht bald auf Denkfehler aufmerksam machen können. Wir empfehlen Ihnen deshalb, jeweils nur die Lösungen zu einem Lehrbrief an uns einzusenden.

Die Lösungen sind stets in der Reihenfolge zu bringen, in der die Aufgaben im Lehrbrief stehen. Bei Rechenaufgaben soll der ganze Rechnungsgang gebracht und das Endergebnis unterstrichen werden. Benutzen Sie dabei aber bitte keinen Rotstift, weil Rot die Korrekturfarbe ist. Lassen Sie bitte zwischen den einzelnen Lösungen mindestens 1 cm Platz frei.

Achten Sie bitte auf die vollständige Lösung und Einsendung aller Aufgaben! Lückenhafte Arbeiten müssen unkorrigiert zurückgesandt werden.

Sie können nur dann ein Abschlußzeugnis über die erfolgreiche Teilnahme am Lehrgang erhalten, wenn sämtliche Aufgabenlösungen zur Korrektur vorgelegt wurden.

Die einzelnen Blätter sind mit Briefklammern zusammenzuheften, wobei – bei der Einsendung mehrerer Lösungen – die Lösungen zur niedrigsten Lehrbriefnummer oben liegen sollen. Wenn Sie die Lösungen mehrerer Lehrbriefe zugleich einsenden, dann stecken Sie bitte an das oberste Blatt der Sendung einen kleinen Zettel, auf den Sie z. B. schreiben: „Lösungen zu den Lehrbriefen 2 bis 4".

Im übrigen vermeiden Sie bitte, kleine Zettel beizulegen. Auch Mitteilungen auf Postabschnitten usw. sowie Fragen, die zwischen die Aufgabenlösungen eingestreut sind, werden leicht übersehen. Für Fragen zum Lehrstoff legen Sie bitte ein besonderes A4-Blatt bei, das Sie mit Namen und Studiennummer versehen.

Die Rücksendung der korrigierten Lösungen mit dem nächstfolgenden Lehrbrief ist mit Rücksicht auf die Arbeitseinteilung am Institut nicht möglich; der Versand der Lehrbriefe erfolgt unabhängig von den korrigierten Lösungen.

Sehen Sie die korrigierten Lösungen genau durch! Finden Sie eine unklare Korrektur, dann senden Sie diese mit einem entsprechenden Vermerk und sämtlichen zu diesem Lehrbrief gehörenden Lösungsblättern mit den folgenden Lösungen nochmals ein! Wir sind Ihnen dankbar, wenn Sie uns auf ein Versehen aufmerksam machen.

Die Einsendung der Aufgabenlösungen muß innerhalb von 5 Jahren erfolgen, vom Beginn des Studiums an gerechnet.

Wenn Sie diese Hinweise beachten, dann ist es uns möglich, Ihre Arbeiten in kürzester Zeit durchzusehen und zurückzusenden.

Prüfungsaufgaben

Lehrbrief 1

Bitte beachten Sie beim Anfertigen und Einsenden der Lösungen zu diesen Prüfungsaufgaben die allgemeinen Hinweise auf den Seiten P1 und P2.

1. a) Für die Bit-serielle, Byte-serielle Übertragung von Daten von einem System zur Peripherie werden drei Leitungen benötigt: Die Signal-Leitung zur Übertragung der Bits, die CTS-Leitung, auf der die Peripherie den Empfang von Daten quittiert, und eine gemeinsame Masse-Leitung. (Vgl. Seite H4.)

   Bitte geben Sie an, welcher Signalpegel auf der Signal-Leitung steht, wenn die serielle Schnittstelle aktiviert ist, aber im Augenblick keine Daten übertragen werden.

   b) Welches Bit eines Bytes wird bei der Bit-seriellen, Byte-seriellen Datenübertragung jeweils als erstes übertragen?

   Bitte geben Sie nur die Nummer dieses Bits an.

2. Nach der Beschreibung der Z80 PIO werden wir Ihnen in diesem Lehrgang einen Timer vorstellen, der als CTC bezeichnet wird. (Vgl. Seite H8.) Was es mit diesem Baustein auf sich hat, braucht Sie hier noch nicht zu interessieren. Hier wollen wir nur feststellen, daß dieser Peripherie-Baustein in ganz ähnlicher Weise adressiert wird wie die Z80 PIO. Anstelle der PIO-Anschlüsse B/$\overline{A}$ und C/$\overline{D}$ hat der Timer die Anschlüsse CSØ und CS1, die entsprechende Funktionen haben. Außerdem hat er einen Anschluß $\overline{CE}$ mit genau der gleichen Funktion wie der Anschluß $\overline{CE}$ der PIO.

   Sehen Sie sich bitte das Bild H19.1 an! Dort ist am Anschluß Y1 des Adreß-Decoders 74 LS 139 eine Leitung eingetragen, die zum Anschluß $\overline{CE}$ des Timers CTC führt.

   Geben Sie bitte an, über welche Port-Adressen der CPU der Timer CTC adressiert werden kann!

   (Die Adressen-Decodierung ist auch beim Timer CTC unvollständig. Es genügt, wenn Sie vier Port-Adressen angeben.)

3. Auf der Seite S8 wurde gezeigt, welche Befehle der Z80 PIO übermittelt werden müssen, wenn der Schnittstellen-Baustein in einer der Betriebsarten 0, 1, 2 oder 3 arbeiten soll.

   Die Interrupt-Programmierung werden wir Ihnen in den folgenden Lehrbriefen vorstellen. Von den Möglichkeiten, die diese Programmierung bietet, sei deshalb hier zunächst abgesehen.

   a) Wieviele **Befehls-Bytes** müssen der Z80 PIO übermittelt werden, wenn einer der Ports A oder B in der Betriebsart 1 arbeiten soll?

b) Wieviele **Befehls-Bytes** müssen der Z 80 PIO übermittelt werden, wenn einer der Ports A oder B in der Betriebsart 3 arbeiten soll?

4. Geben Sie bitte die Befehlsfolge an, mit welcher die Z 80 PIO wie folgt programmiert wird:

   Der PIO-Port A soll in der Betriebsart 1 arbeiten.

   Der PIO-Port B soll in der Betriebsart 3 arbeiten.

   Die Leitungen 3 und 4 des Ports B sollen als Ausgänge (Sender) programmiert werden, alle anderen Leitungen des Ports B sollen als Eingänge (Empfänger) programmiert werden.

Prüfungsaufgaben

Lehrbrief 2

Bitte beachten Sie beim Anfertigen und Einsenden der Lösungen zu diesen Prüfungsaufgaben die allgemeinen Hinweise auf den Seiten P1 und P2.

1. An ein Mikroprozessor-System soll ein Drucker mit Centronics-Schnittstelle angeschlossen werden. Im System ist eine Z 80 PIO verfügbar, über deren Port B jedoch bereits an den Anschlüssen 0 bis 4 eine Daten-Kommunikation mit einer hier nicht interessierenden, anderen Peripherie abgewickelt wird.

   Für die Centronics-Schnittstelle stehen die acht Anschlüsse des Ports A und die Anschlüsse 5 bis 7 des Ports B zur Verfügung.

   Weil die Anzahl der verfügbaren Port-Anschlüsse begrenzt ist, soll bei der Drucker-Schnittstelle auf die Auswertung des Paper-Out-Signals (vgl. Seite H27) verzichtet werden.

   Das Bild P5.1 zeigt, wie der Drucker angeschlossen werden soll. Beachten Sie bitte die Signal-Richtungen für die Daten-Kommunikation mit der anderen Peripherie.

   Tragen Sie bitte auf dem Lösungsblatt eine Initialisierungs-Routine zur Initialisierung der PIO ein. Schreiben Sie diese Routine in der gleichen Form an wie die entsprechende Routine auf der Seie H29. Geben Sie für die *I/O Register Control Words* die Bitmuster der Bytes an, und vergessen Sie nicht, diese Bitmuster durch den Buchstaben B zu kennzeichnen (vgl. Seite L2). – Ergänzen Sie bitte die Kommentare im Anschluß an die Strichpunkte!

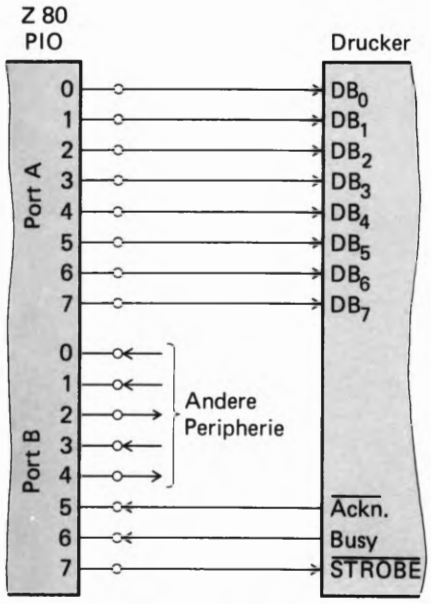


Bild P5.1
Zu Prüfungsaufgabe 1: Die PIO-Ports A und B sollen so programmiert werden, daß die hier dargestellte Centronics-Schnittstelle realisiert wird.

2. In der *Daisy Chain* der Peripherie-Bausteine des Micro-Professors sind nur der CTC-Baustein und die Z 80 PIO angeordnet (vgl. das Bild H41).

   Tragen Sie bitte in die Zeichnung auf dem Lösungsblatt die Werte der IEI- und IEO-Signale für den Fall ein, daß von der CPU gerade ein von der PIO angeforderter Interrupt bearbeitet wird. Vom CTC-Baustein liegt keine Interrupt-Anforderung vor.

3. Im Morse-Programm mit Interrupt-Möglichkeit (Programm ab der Seite L31) können die ALARM- und SOS-Interrupts jeweils durch eins von zwei ODER-verknüpften Signalen angefordert werden. Diese Eigenschaft ist durch die *Interrupt Control Words* bei der Programmierung des Ports A und des Ports B bestimmt.

   a) Geben Sie bitte an, welche Programmänderung Sie vornehmen müssen, wenn das Alarm-Signal nur dann ausgelöst werden soll, wenn über die Anschlüsse Nr. 0 und Nr. 1 des Ports A gleichzeitig Interrupt-Anforderungen vorliegen.

   b) Wie müssen Sie das Programm ändern, wenn die Signale an den Anschlüssen Nr. 6 und Nr. 7 des Ports B normalerweise die

Werte 1 haben und wenn ein SOS-Interrupt dann ausgelöst werden soll, wenn eins dieser Signale auf den Wert 0 wechselt?

Sie brauchen für Ihre Lösungen jeweils nur die Adresse im Programm anzugeben, bei der ein Byte geändert werden muß, und daneben das neue Byte anzuschreiben.

4. Im Morse-Programm mit Interrupt-Möglichkeit soll der Alarm durch die Betätigung der zum Port A gehörenden Taste Nr. 4 und die Ausgabe des SOS durch die Betätigung der zum Port B gehörenden Taste Nr. 3 ausgelöst werden. Die Betätigung aller anderen Tasten soll wirkungslos bleiben.

   a) Schreiben Sie bitte die Adressen in der INIT-Routine an, bei denen die Bytes geändert werden müssen, damit die eben angegebene Funktion realisiert wird. — Tragen Sie neben diesen Adressen die Werte der neuen Bytes ein.

      Wenn Sie Ihre Lösung in einem Versuch ausprobieren, dann beachten Sie bitte, daß die steckbaren Brücken für die Tasten der Ports A und B umgesteckt werden müssen! (Vgl. die Bilder H 12.1 und H 14.1.)

   b) Begründen Sie bitte, warum die beiden unterschiedlichen Interrupts nicht über zwei Tasten ein und desselben PIO-Ports ausgelöst werden können!

5. Auf der Seite S 40 haben wir festgestellt und begründet, warum zwar bei entsprechend programmierten EI-Befehlen die Interrupt-Service-Routine SOS durch einen ALARM-Interrupt unterbrochen werden kann, umgekehrt aber eine Unterbrechung der ALARM-Routinen durch ein SOS-Interrupt nicht möglich ist. (Lesen Sie die entsprechenden Abschnitte bitte noch einmal sorgfältig nach!)

   Wenn vorausgesetzt wird, daß in der ALARM-Routine der EI-Befehl gleich am Anfang programmiert ist, dann gibt es eine ganz einfache Möglichkeit, die ALARM-Routine durch die SOS-Routine unterbrechbar zu machen: Ziemlich am Ende des Programms brauchen nur zwei Bytes geändert zu werden.

   Wir geben Ihnen einen Hinweis: Wenn Sie die Änderung vornehmen (Sie können sie ausprobieren!), dann kann die SOS-Routine nicht mehr durch die ALARM-Routine unterbrochen werden.

   a) Durch welche Maßnahme wird es möglich, die ALARM-Routine durch die SOS-Routine zu unterbrechen?

   b) Schreiben Sie bitte die beiden Adressen an, bei denen andere Bytes eingetragen werden müssen, und schreiben Sie diese beiden Bytes jeweils neben die Adresse.

Prüfungsaufgaben

Lehrbrief 3

Bitte beachten Sie beim Anfertigen und Einsenden der Lösungen zu diesen Prüfungsaufgaben die allgemeinen Hinweise auf den Seiten P1 und P2.

1. Im Gegensatz zur Centronics-Druck-Routine auf der Seite L20 haben wir in der Druck-Routine mit Interrupt auf der Seite L39 darauf verzichtet, das zu druckende ASCII-Byte während der Ausgabe des STROBE-Signals im C-Register zu verwahren (vgl. Seite H49). Das ASCII-Byte steht also nach der Ablieferung an den Drucker nicht mehr zu weiterer Verwendung zur Verfügung.

   Wenn man Wert darauf legt, das ASCII-Byte auch nach dem Abarbeiten der Druck-Routine im Akkumulator verfügbar zu haben, dann muß vor der Ausgabe des STROBE-Signals ein LD C,A-Befehl programmiert werden und nach der Ausgabe des STROBE-Signals ein LD A,C-Befehl, der das ASCII-Byte in den Akkumulator zurückholt.

   Wenn diese beiden Befehle in die Druck-Routine eingebaut werden, dann wird der ganze Rest des Programms um die Länge dieser beiden Befehle verschoben.

   Geben Sie bitte an, bei welcher Adresse in diesem Fall das Vektorfeld für die Ablage der Anfangs-Adresse der Interrupt-Service-Routine beginnen muß!

   Ein Hinweis: Wenn Sie diese Programmierung ausprobieren, dann beachten Sie bitte, daß sich die Anfangs-Adresse der INIT-Routine verschiebt. Der CALL INIT-Befehl im Hauptprogramm muß also entsprechend geändert werden. Ebenfalls ändert sich beim Label VWAIT das LOB der Adresse WAIT.

2. An welcher Stelle des Interrupt-Lauflicht-Programms ab der Seite L47 müssen Sie eine Änderung vornehmen, wenn das Lauflicht langsamer laufen soll?

   Begründen Sie bitte Ihre Antwort.

3. Geben Sie bitte die vollständige Programmierung des CTC-Kanals Ø zur Lösung folgender Aufgabe an:

   Kein Software-Reset. Der CTC soll bei jedem fünfundzwanzigsten 1-Ø-Signal, das ihm von der Peripherie übermittelt wird, die Bearbeitung einer Interrupt-Service-Routine veranlassen. Die Anfangs-Adresse der Interrupt-Service-Routine ist im Vektorfeld abgelegt.

   Gehen Sie davon aus, daß die CPU bereits zur Verarbeitung von Interrupts programmiert worden ist.

   Das letzte Befehls-Byte des Programms, zu dem die CTC-Programmierung gehört, steht bei der Adresse 1893H.

4. Im Zusammenhang mit dem Programm zur Interrupt-gesteuerten, seriellen Daten-Ausgabe haben wir auf der Seite S64 angegeben, daß im Versuchs-Programm die Ausgabe der einzelnen Bits im Abstand von etwa einer halben Sekunde erfolgt.

   Das Bild S65.1 zeigt das Prinzip, nach dem der zeitliche Abstand der Bit-Ausgabe durch die Programmierung der CTC-Kanäle bestimmt wird. – Sehen Sie sich bitte die CTC-Programmierung auf der Seite L61 an!

   Wie lang ist die Zeit genau, die zwischen der seriellen Ausgabe von zwei Bits liegt?

5. Das ab der Seite L59 angegebene Programm zur Interrupt-gesteuerten, seriellen Ausgabe von Bytes kann auch praktisch eingesetzt werden. Wir haben auf der Seite S65 darauf hingewiesen, daß dazu die Teile des Programms weggelassen werden können, mit denen das Bitmuster der ausgegebenen Bytes über die zum PIO-Port A gehörenden Leuchtdioden angezeigt wird.

   Bei der praktischen Verwendung wird auch eine höhere Ausgabe-Geschwindigkeit gewählt. In der Darstellung des Bilds S65.1 kann auf die Verwendung des CTC-Kanals 1 verzichtet werden.

   a) Geben Sie bitte das *Channel Control Word* für den CTC-Kanal Ø an, wenn dieser allein die Interrupt-Anforderungen für die SEROUT-Routine (Seite L63) liefern soll.

   b) Geben Sie bitte das *Time Constant Word* für den CTC-Kanal Ø an, wenn seine Interrupt-Anforderungen in solchem zeitlichen Abstand erfolgen sollen, daß die serielle Daten-Ausgabe mit 300 Baud (300 Bit/s) erfolgt.

Prüfungsaufgaben

Lehrbrief 4

Bitte beachten Sie beim Anfertigen und Einsenden der Lösungen zu diesen Prüfungsaufgaben die allgemeinen Hinweise auf den Seiten P1 und P2.

1. Der Ein- und Ausgabebaustein 8255 kann in drei verschiedenen Betriebsarten arbeiten. Welche Betriebsart arbeitet ohne Steuersignale; also im reinen Ein- Ausgabebetrieb?

2. Der 8255 wird durch das Einschreiben eines Steuerworts in das Steuerwortregister des Bausteins programmiert. Das einmal in den Baustein eingeschriebene Steuerwort hat solange Gültigkeit, bis es durch ein neues Steuerwort überschrieben wird oder die Spannungsversorgung ausfällt.

   Angenommen, dem 8255 wird die Sedezimalzahl AFH in das Steuerwortregister eingeschrieben. Welche Funktion haben durch dieses Steuerwort der Port A und der Port B des 8255?

3. Gerade weil der 8255 so universell programmierbar ist, kann er für die unterschiedlichsten Aufgaben eingesetzt werden. Es ist sogar möglich, daß zwei verschiedene Ports in unterschiedlichen Betriebsarten arbeiten.

   Geben Sie bitte die Befehlsfolge an, mit welcher der 8255 wie folgt programmiert wird:

   Der Port A arbeitet in der Betriebsart 1 als Ausgabeport

   Der Port B arbeitet in der Betriebsart 0 als Eingabeport

   Die an Port C nicht für Steuersignale gebrauchten Bits arbeiten als Ausgabebits.

   Das interne INTE-Flipflop, das für Port A zuständig ist, soll gesetzt sein.

4. Im Fachgebiet Hardware dieses Lehrbriefs haben Sie gelesen, welche Steuersignale notwendig sind, um ein Interrupt-Signal an Port C zu erhalten, wenn der Baustein in der Betriebsart 1 arbeiten soll.

   Bitte geben Sie an, durch welche Signale ein Interrupt-Signal entsteht, wenn dem Baustein das Steuerwort 1Ø11 11Ø1 gegeben wurde.

5. Wenn der Ein- Ausgabebaustein 8255 in der Betriebsart 2 arbeitet, stehen verschiedene Steuersignale für den Datenverkehr über Port A zur Verfügung. Dennoch bleiben am Port C Bits, die dem Anwender als Aus- oder Eingabebits zur Verfügung stehen.

   a) Welche Bits werden in der Betriebsart 2 nicht als Steuersignale verwendet? (Gemeint sind die freibleibenden Bits an Port C).

   b) Welche Steuersignale werden für die Dateneingabe über Port A betätigt?

   c) Welche Steuersignale werden für die Datenausgabe über Port A betätigt?

Lehrgang PERIPHERIE-BAUSTEINE Verfasser: Hans Fischer, Edgar Hoch
Herausgeber: R. Christiani

Übungen

Lösungen der im Text gestellten Aufgaben

Bitte sehen Sie sich die folgenden Lösungen erst dann an, wenn Sie die im Text gestellten Aufgaben selbständig durchgearbeitet haben.

Zu Aufgabe S 6.1

Welche Form der Befehl OUT (PORT),A haben muß, damit das mit dem Befehl LD A,AFH in den Akkumulator geladene Byte so an die Z 80 PIO geschickt wird, daß es richtig als Befehls-Byte interpretiert wird, zeigen wir Ihnen im Kapitel über die Adressierung der PIO.

Das in den Akkumulator geladene Befehls-Byte AFH hat das Bitmuster

1 Ø 1 Ø 1 1 1 1

Die vier niederwertigen 1-Bits zeigen der PIO, daß es sich bei dem Befehls-Byte um ein Betriebsart-Steuerwort *(Mode Word)* handelt.

Entsprechend der Darstellung im Bild S 6.1 sind im Betriebsart-Steuerwort nur die hier verstärkt gedruckten Bits maßgeblich. Die Bits Nr. 5 und Nr. 4 haben *Don't Care*-Charakter und können beliebige Werte annehmen. Im Beispiel zur Aufgabe haben sie die Werte 1 und 0, haben aber keinen Einfluß auf die Wirkung des Betriebsart-Steuerworts.

Maßgeblich für die Wirkung des Betriebsart-Steuerworts sind die Bits Nr. 7 und Nr. 6, welche die Werte 1 und 0 haben und damit die Betriebsart 2 kennzeichnen.

Die beiden Befehle steuern den vom OUT-Befehl angesprochenen Port der Z 80 PIO in die Betriebsart 2.

Entsprechend der Arbeitsweise dieser Betriebsart muß mit dem OUT-Befehl der PIO-Port A angesprochen werden.

Zu Aufgabe H 18.1

In der Aufgabenstellung wurde ausdrücklich darauf hingewiesen, daß bei der genannten Befehlsfolge der PIO-Anschluß $\overline{CE}$ zunächst unberücksichtigt bleibt. Die im Anschluß an die Aufgabe folgenden Erläuterungen zeigen, daß die Befehlsfolge in der dargestellten Form tatsächlich bei der PIO noch keine Wirkung hat.

Läßt man den Anschluß $\overline{CE}$ außer Betracht, dann gilt die Überlegung:

Das Bitmuster Ø 1 1 Ø 1 1 1 1 des Bytes 6FH im B-Register ist das Steuerwort für die Betriebsart 1.

Mit dem 3. Befehl der genannten Befehlsfolge (OUT (C),B) wird das Byte aus dem B-Register über den CPU-Port ausgegeben, dessen Adresse im C-Register abgelegt ist. (Vgl. Lehrgang Mikroprozessortechnik, Seite H 64.)

Da mit dem zweiten Befehl der Befehlsfolge im C-Register die Port-Adresse Ø3 abgelegt wird, wird das Betriebsart-Steuerwort aus dem B-Register der PIO als Befehl für den Port B übermittelt.

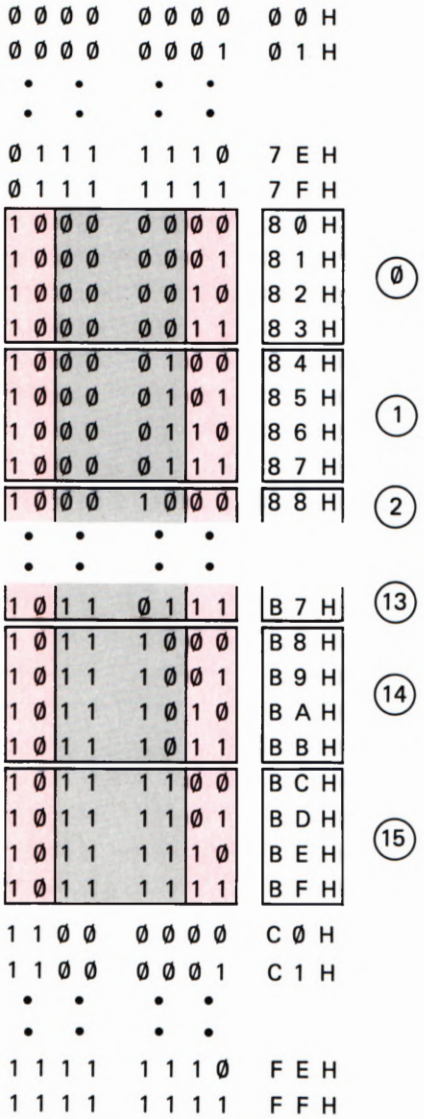


Bild Ü2.1
Zu Aufgabe H22.1: Für die Adressierung der Z80 PIO werden 16 x 4 = 64 Port-Adressen belegt.

Zu Aufgabe H22.1

Sehen Sie sich das Bild Ü2.1 an! Wir haben das Prinzip der Bitmuster für die 256 möglichen Port-Adressen dargestellt.

Wegen der vier *Don't-Care*-Bits in den unvollständig decodierten Adressen kommen die auf der Seite H21 stärker gedruckten Kombinationen der Adressen-Bits Nr. 7 und Nr. 6 sowie Nr. 1 und Nr. 0 innerhalb der 256 möglichen Port-Adressen insgesamt sechzehnmal vor.

Einmal werden die vier Port-Adressen benötigt (80H bis 83H). Weitere 15 mal 4 = 60 Adressen, die die gleichen Kombinationen der Bits Nr. 7, Nr. 6, Nr. 1 und Nr. 0 enthalten, werden für anderweitige Verwendung unbrauchbar.

Das Bild Ü2.1 macht deutlich, daß diese unbrauchbar gewordenen Adressen von 84H bis BFH einschließlich reichen.

Zu Aufgabe S7.1

	Mnemonische Codes	Befehls-Bytes
a)	LD A,0CFH	3E CF
	OUT (83H),A	D3 83
b)	LD A,0FFH	3D FF
	OUT (0BFH),A	D3 BF

Beachten Sie bitte, daß wir bei den mnemonischen Codes die Regel angewandt haben, die Befehls-Operanden immer mit einer der Ziffern 0 bis 9 beginnen zu lassen. Außerdem haben wir die Operanden mit dem nachgestellten Buchstaben H als sedezimal (**H**exadezimal) dargestellt gekennzeichnet.

Zu Aufgabe S9.1

a)	Menmonische Codes	b)	Befehls-Bytes
	LD A,0CFH		3E CF
	OUT (82H),A		D3 82
	LD A,10010100B		3E 94
	OUT (82H),A		D3 82

Bei der Programmierung des I/O-Register-Steuerworts haben wir beim Anschreiben der mnemonischen Codes den Operanden des LD A-Befehls, also das an den Befehls-Port der PIO zu sendende I/O-Register-Steuerwort, durch den nachgestellten Buchstaben B als Binär dargestellt gekennzeichnet.

Zu Aufgabe S18.1

Das Bild H12.1 zeigt, an welche Anschlüsse des PIO-Ports A die Relais angeschlossen sind.

Im Programm zum Versuch S15.1 wird das Relais am Anschluß 0 des Ports A angesteuert, wenn der Umlauf des Lichtpunkts gestartet wird. Das Relais am Anschluß 1 des Ports A wird bei jedem Umlauf des Lichtpunkts angesteuert. (Vgl. das Bild S14.2.)

Lösungen der im Text gestellten Aufgaben

Bitte sehen Sie sich die folgenden Lösungen erst dann an, wenn Sie die im Text gestellten Aufgaben selbständig durchgearbeitet haben.

Zu Aufgabe H29.1

Zweckmäßig werden wieder beide Ports für die Betriebsart 3 programmiert.

Für die Belegung der Anschlüsse des Ports A entsprechend der Aufgabenstellung ergibt sich das im Bild Ü3.1 dargestellte Bitmuster des I/O-Register-Steuerworts: Die Bits Nr. 0, 1 und 2 haben die Werte 1 zur Programmierung der Anschlüsse für die Signale Paper Out, Acknowledge und Busy als Eingänge (Empfänger). Mit dem Wert Ø des Bits Nr. 3 wird der Anschluß für das STROBE-Signal als Ausgang (Sender) programmiert. – Die Bits Nr. 4, 5, 6 und 7 haben *Don't Care*-Werte.

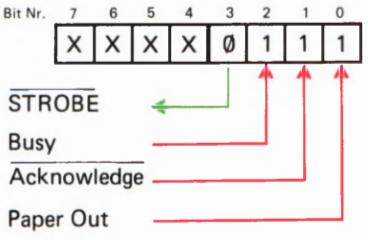


Bild Ü3.1
Zu Aufgabe H29.1: Darstellung des Bitmusters in I/O-Register-Steuerwort.

Da sämtliche Anschlüsse des Ports B als Ausgänge (Sender) für die acht Bits eines Bytes arbeiten, gehört zum Port B das I/O-Register-Steuerwort ØØ.

Diese Überlegungen führen zu folgender Routine für die Programmierung der PIO:

```
INIT: LD   A,ØCFH        ;Port A: Betriebsart 3
      OUT  (PIOBA),A
      LD   A,11110111B   ;        Bit Nr. Ø Paper Out Eingang
                         ;        Bit Nr. 1 Ackn.    Eingang
                         ;        Bit Nr. 2 Busy     Eingang
                         ;        Bit Nr. 3 STROBE   Ausgang
      OUT  (PIOBA),A
      LD   A,ØCFH        ;Port B: Betriebsart 3
      OUT  (PIOBB),A
      LD   A,ØØØØØØØØB   ;        Alle Anschlüsse Ausgänge
      OUT  (PIOBB),A
      RET
```

Zu Aufgabe H34.1

Das Bild Ü3.2 zeigt oben das Bitmuster im Akkumulator nach der Ausführung der Befehle IN A,(PIODB) und AND 7ØH. Die Werte der Handshake-Signale haben wir rot eingetragen.

Nach der bitweisen exklusiv-ODER-Verknüpfung des Akkumulator-Inhalts mit dem Operanden 1ØH des XOR-Befehls steht im Akkumulator ein Byte, in dem das Bit Nr. 6 den Wert 1 hat.

Der Inhalt des Akkumulators ist ungleich ØØ; das DRUCK-Programm kann die Abfrage-Schleife für den Drucker-Status nicht verlassen.

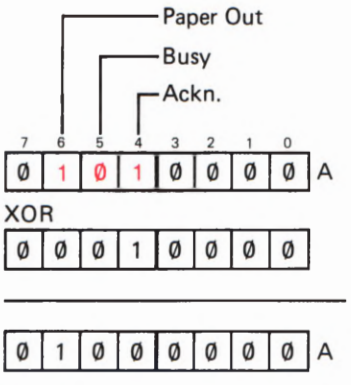


Bild Ü3.2
Zu Aufgabe H34.1: Bei den rot eingetragenen Bit-Werten im Akkumulator ergibt sich nach der Ausführung des XOR-Befehls ein von ØØ abweichendes Byte.

Zu Aufgabe S20.1

Die Auflistung des geänderten Programms finden Sie ab der Seite L 27. Sie werden als erstes feststellen, daß dieses Programm ganz offensichtlich kürzer ist als das ursprüngliche Lauflicht-Programm. Keine Sorge: Wir werden Ihnen noch zeigen, daß sich der zunächst scheinbar größere Aufwand im ursprünglichen Programm jedenfalls rentiert.

Im Programm haben wir die geänderten Anweisungen mit kleinen Buchstaben eingetragen. Sehen Sie sich zunächst das Hauptprogramm an.

Nach dem Befehl OUT (PIODB),A beim Label LOOP zur Ausgabe des Bitmusters für die Leuchtdioden werden mit dem Befehl IN A,(PIODA) die von der Peripherie über den PIO-Port A gelieferten Signale abgefragt. Solche Signale werden nur über die Bit-Anschlüsse Ø und 1 dieses Ports erwartet. Der anschließende Befehl AND ØØØØØØ11B setzt die entsprechende Maske.

Weil der IN A,(PIODA)-Befehl das später noch benötigte Leuchtdioden-Bitmuster im Akkumulator überschreibt, wird dieses Bitmuster mit dem PUSH AF-Befehl auf den Stack gerettet. Von dort wird es vor dem Rotationsbefehl RLCA mit dem Befehl POP AF in den Akkumulator zurückgeholt.

Vom Wert der mit den Port-A-Tasten Nr. 0 und Nr. 1 gelieferten Peripherie-Signale hängt es ab, ob die ALARM-Routine aufgerufen wird oder nicht. Diese Entscheidung trifft der CALL NZ,ALARM-Befehl. Er ruft die ALARM-Routine nur dann auf, wenn nach der Maskierung im Akkumulator ein von einer Taste geliefertes 1-Bit entdeckt wird. Andernfalls stehen im Akkumulator nur Ø-Bits, und der CALL-Befehl wird ignoriert.

Beachten Sie bei der Eingabe des Programms bitte die Änderungen, die sich gegenüber dem ursprünglichen Programm ergeben! Bei der Adresse 18Ø1H ändert sich das LOB der Adresse der INIT-Routine. Von der Adresse 18Ø7H bis einschließlich der Adresse 1826H muß das Programm neu eingetastet werden. Bei den Adressen 1826H und 1853H müssen Sie jeweils das Byte C9 für den RET-Befehl eintasten. Die übrigen Teile des Programms können unverändert bleiben.

Zu Aufgabe S24.1

a) Bei unveränderten Adressen der Ablage-Speicherzellen muß in der Speicherzelle mit der Adresse 1858H das LOB F7 und in der folgenden Speicherzelle das HOB 18H der Startadresse der ALARM-Routine stehen.

b) Das HOB 19H der niedrigeren der beiden Ablage-Adressen muß im I-Register der CPU eingetragen sein. Das LOB 26H dieser Ablage-Speicherzelle liefert der Interrupt-Vektor der Z 80 PIO.

c) Wenn die erste der Ablage-Speicherzellen die Adresse 1925H hat, dann muß bei einer Interrupt-Anforderung die PIO das LOB 25H mit dem Bitmuster ØØ1Ø Ø1Ø1 dieser Adresse liefern. In diesem Bitmuster hat das Bit Nr. 0 den Wert 1, und das ist nicht zulässig. Die CPU kann die Interrupt-Service-Routine nicht finden.

Zu Aufgabe S31.1

Die Befehlsbytes für den PIO-Port A werden nacheinander zunächst in den Akkumulator geladen und dann mit OUT-Befehlen an die Befehlsadresse PIOBA (beim Micro-Professor = 82H) geliefert.

Bei der Reihenfolge der Befehle halten Sie sich zweckmäßig an die Aufstellung auf der Seite S8.

Die in der Aufgabenstellung gewünschte Interrupt-Programmierung ist nur in der Betriebsart 3 der PIO möglich. Das dafür zuständige *Mode Control Word* ist CF mit dem Bitmuster 11ØØ 1111 (Bild S6.1).

Im anschließenden *I/O Register Control Word* haben die Bits Nr. 0, Nr. 2 und Nr. 7 die Wert 1; alle übrigen Bits erhalten die Werte Ø. Es ergibt sich das Bitmuster 1ØØØ Ø1Ø1 mit dem Byte 85H (Bild S9.1).

Im Lauflicht-Programm ist das LOB der Interrupt-Service-Routine bei der Adresse 18**58** abgelegt. Sie können davon ausgehen, daß das HOB dieser Adresse bereits im I-Register der CPU steht. Das LOB 58H der Ablage-Adresse wird der PIO als *Interrupt Vector Word* übermittelt (Bild S22.1).

Im *Interrupt Control Word* (Bild S28.1) wird der PIO mit dem Wert 1 des Bits Nr. 4 mitgeteilt, daß anschließend ein *Mask Control Word* ausgegeben wird. – Weil ein Interrupt bei Betätigung der Tasten, also durch 1-Signale ausgelöst werden soll, hat das Bit Nr. 5 den Wert 1. – Die Forderung nach Auslösen des Interrupts bei gleichzeitiger Betätigung mehrerer Tasten bedingt eine UND-Verknüpfung der entsprechenden Interrupt-Signale. Das Bit Nr. 6 erhält also den Wert 1. – Das Bit Nr. 7 muß den Wert 1 haben, wenn die Interrupt-Anforderungen wirksam werden sollen. – Insgesamt ergibt sich das Byte F7 mit dem Bitmuster 1111 Ø111.

Das *Mask Control Word* ernennt die vorher als Eingänge programmierten Port-Anschlüsse Nr. 0, Nr. 2 und Nr. 7 mit Werten Ø bei den entsprechenden Bitnummern zu Interrupt-Eingängen (Seite S31). Es ergibt sich das Bitmuster Ø111 1Ø1Ø für das Byte 7A.

Weil Sie das Ergebnis der Aufgabenlösung am Beispiel des Lauflicht-Programms ausprobieren können, setzen wir in der anschließenden Befehlsauflistung die Adressen ein, bei denen die Initialisierung des Ports A eingetragen wird. Die zu ändernden Bytes haben wir stärker gedruckt.

```
1815  3E  CF   LD   A,11ØØ1111B  ;Mode Control Word
1817  D3  82   OUT  (PIOBA),A

1819  3E  85   LD   A,1ØØØØ1Ø1B  ;I/O Register Control Word
181B  D3  82   OUT  (PIOBA),A

181D  3E  58   LD   A,58H        ;Interrupt Vector Word
181F  D3  82   OUT  (PIOBA),A

1821  3E  F7   LD   A,1111Ø111B  ;Interrupt Control Word
1823  D3  82   OUT  (PIOBA),A

1825  3D  7A   LD   A,Ø1111Ø1ØB  ;Mask Control Word
1827  D3  82   OUT  (PIOBA),A
```

Zu Aufgabe H 40.1

Nach dem Einschalten ist das Interrupt-Flipflop IFF1 in der CPU zunächst zurückgesetzt. Die CPU verweigert die Annahme von Interrupt-Anforderungen.

In der INIT-Routine, die im Lauflicht-Programm als erstes aufgerufen wird, setzt EI bei der Adresse 182F das Interrupt-Flipflop.

Ersetzen Sie im Programm versuchsweise das Byte FB bei der Adresse 182F durch das Byte ØØ für einen NOP-Befehl! Jetzt bleibt das Interrupt-Flipflop nach dem Start des Programms zurückgesetzt; Interrupt-Anforderungen bleiben unberücksichtigt. Sie können keinen Alarm auslösen.

Zu Aufgabe H 42.1

a) Die Quittung von der CPU beweist, daß die PIO nicht durch ein Ø-Signal an ihrem IEI-Anschluß an der Weitergabe der Interrupt-Anforderung gehindert wurde. Solange der CTC-Baustein seinerseits keine Interrupt-Anforderung abschickt, bleibt das IEI-Signal der PIO auf dem Wert 1.

   Durch die Quittung der Interrupt-Anforderung erscheint am IEO-Anschluß der PIO ein Ø-Signal. Alle in der *Daisy Chain* folgenden Peripherie-Bausteine werden dadurch für die Weitergabe von Interrupt-Anforderungen gesperrt.

b) Der CTC-Baustein steht in der *Daisy Chain* an erster Stelle. An seinen IEI-Anschluß ist mit +5 V ein festes 1-Signal geschaltet. Er ist also grundsätzlich zur Weitergabe von Interrupt-Anforderungen ermächtigt, unabhängig davon, ob ein in der *Daisy Chain* folgender Baustein bereits eine Interrupt-Anforderung weitergegeben hat.

   Wenn der CTC-Baustein während der Bearbeitung einer anderen Interrupt-Service-Routine eine Interrupt-Anforderung weitergibt, dann hängt es von der Stellung des EI-Befehls in der Routine ab, ob die CPU dieser Anforderung nachkommt oder nicht.

   Steht der EI-Befehl am Ende der gerade in Arbeit befindlichen Interrupt-Service-Routine, dann kommt die Interrupt-Anforderung vom CTC-Baustein zwar bei der CPU an, sie wird jedoch nicht ausgeführt.

   Steht der EI-Befehl gleich als erster in der in Arbeit befindlichen Interrupt-Service-Routine, dann wird diese Routine durch den neuerlichen Interrupt nochmals unterbrochen. Das ist aber nur möglich, weil der CTC-Baustein wegen seiner vorrangigen Stellung in der *Daisy Chain* die Interrupt-Anforderung weitergeben konnte.

   Wenn der „interruptende" Interrupt von der CPU quittiert wird, dann schaltet der CTC-Baustein sein IEO-Signal auf den Wert Ø. Damit steht am IEI-Anschluß der PIO ein Ø-Signal. Am IEO-Anschluß der PIO erscheint erst dann wieder ein 1-Signal zur Freigabe der folgenden Peripherie-Bausteine, wenn sowohl der CTC-Baustein ein IEO-1-Signal liefert, als auch die von der PIO aufgerufene Interrupt-Serivce-Routine abgeschlossen worden ist.

Lösungen der im Text gestellten Aufgaben

Bitte sehen Sie sich die folgenden Lösungen erst dann an, wenn Sie die im Text gestellten Aufgaben selbständig durchgearbeitet haben.

Zu Aufgabe H48.1

Die fertig programmierte INIT-Routine finden Sie auf der Seite L40.

Die Programmierung des PIO-Ports A bedarf keiner Erläuterung. Auch die Programmierung der Betriebsart des Ports B sowie das *I/O Register Control Word* für die Programmierung der PIO-Anschlüsse 5 und 6 als Eingänge ist problemlos.

Zur Übermittlung des *Interrupt Vector Words* an den Befehls-Port PIOBB wird der Akkumulator mit dem LOB des Interrupt-Vektors geladen. Er zeigt auf die Adresse VWAIT. Sie haben bereits auf der Seite L24 gesehen, daß der Assembler das LOB einer Adresse erkennt, wenn es im Operanden des LD A,-Befehls als LOW bezeichnet wird. Der Lade-Befehl hat also die Form LD A,LOW VWAIT.

Im *Interrupt Control Word* gibt der Wert 1 des Bits Nr. 7 die PIO zur Weitergabe von Interrupt-Anforderungen frei. – Der Wert 1 des Bits Nr. 6 bewirkt die UND-Verknüpfung der Signale, die eine Interrupt-Anforderung auslösen. – Die Auslösung von Interrupt-Anforderungen mit LOW-Signalen bewirkt der Wert Ø des Bits Nr. 5. – Mit dem Wert 1 des Bits Nr. 4 wird der PIO mitgeteilt, daß dem *Interrupt Control Word* noch ein *Mask Control Word* folgt. Zusammen mit dem Muster Ø111 der Kennungs-Bits Nr. 3 bis Nr. Ø ergibt sich für das *Interrupt Control Word* das Byte 11Ø1 Ø111 mit der sedezimalen Darstellung ØD7H.

Im *Mask Control Word* haben die Bits die Werte Ø, welche die entsprechenden Port-Anschlüsse für Interrupt-Anforderungen programmieren. So entsteht entsprechend der Aufgabenstellung das Bitmuster 1ØØ11111.

Das HOB 18H des Interrupt-Vektors muß auf dem Wege über den Akkumulator in das I-Register der CPU geladen werden.

Die CPU wird mit dem Befehl IM2 für den Interrupt-Mode 2 programmiert. Anschließend erfolgt mit dem Befehl EI das Setzen des Interrupt-Flipflops IFF1.

Weil die Programmierung in einem Unterprogramm vorgenommen werden soll, darf am Ende der Routine der Befehl RET nicht vergessen werden.

Zu Aufgabe H52.1

Die nächste freie Adresse ist 1947H. Bei dieser Adresse darf das Vektorfeld nicht beginnen, weil das Bit Nr. Ø im LOB der ersten Vektorfeld-Adresse den Wert Ø haben muß (vgl. Seite S23). Das ist erst bei der nächstfolgenden Adresse 1948H der Fall.

Der Interrupt-Vektor für die Interrupt-Service-Routine WAIT muß auf die erste Adresse des Vektorfelds weisen. Das HOB (19H) dieser Adresse wird im I-Register der CPU abgelegt; das LOB (48H) der Adresse wird der PIO über die Port-Adresse PIOBA als *Interrupt Vector Word* übergeben.

Im Programm ergeben sich damit folgende Änderungen:

Adresse	Byte	Funktion
186E	48	Adressen-LOB
1876	19	Adressen-HOB

Zu Aufgabe S47.1

a) Diese Aufgabe ist besonders einfach. – Auf der Seite S47 haben wir beschrieben, daß der zum Kanal Ø gehörende Zähler über das *Time Constant Word* mit der Anzahl der jeweils anzuzählenden Ereignisse geladen wird.

   Auf der Seite L45 sehen Sie, daß das *Time Constant Word* bei der Adresse 18ØA mit dem Befehl LD A,Ø5 in den Akkumulator geladen wird, um anschließend mit einem OUT-Befehl an den Kanal Ø übermittelt zu werden.

   Halten Sie das Programm an, ändern Sie das Byte bei der Adresse 18ØB auf den Wert ØD (= dreizehn) und starten Sie das Programm wieder.

   Machen Sie den Versuch!

b) Auch das ist mehr als einfach. Die steckbare Brücke muß von den mit Ø bezeichneten Stiften des PIO-Ports B auf die mit 6 bezeichneten Stifte umgesteckt werden. Außerdem muß die vom Anschluß ZC/TO0 der Klemmleiste kommende Leitung jetzt an den mit 6 bezeichneten Anschluß geführt werden.

   Diese Änderung ist deshalb so einfach, weil das von der Taste kommende Signal nicht in der PIO verwendet wird. (Die PIO müßte ggf. umprogrammiert werden.)

Zu Aufgabe H58.1

Für die Lösung dieser Aufgabe gelten die gleichen Überlegungen, wie wir sie bei der Lösung der Aufgabe H22.1 auf der Seite Ü2 angestellt haben.

Das Schema für die Port-Adressen, die durch die unvollständige Decodierung belegt werden, können Sie dem Bild Ü2.1 entnehmen, wenn Sie die Werte der beiden Adreß-Bits Nr. 7 und Nr. 6 vertauschen. Auch bei den Werten A7 = Ø und A6 = 1 kommen die auf der Seite H58 stärker gedruckten Kombinationen der Adreß-Bits A7 und A6 sowie A1 und AØ innerhalb der 256 möglichen Port-Adressen insgesamt sechzehnmal vor.

15 mal 4 = 60 Adressen werden nicht benötigt. Sie sind für anderweitige Verwendung unbrauchbar. Diese Adressen reichen von 44H bis einschließlich 7FH.

Zu Aufgabe S52.1

a) Die Bits für das *Channel Control Word* müssen folgende Werte haben:

Bit	Programmierung	Wert
Ø	Kennung des *Channel Control Words*	1
1	Kein Software-Reset	Ø
2	Es folgt ein *Time Constant Word*	1
3	Triggern über den CLK/TRG-Anschluß	1
4	Die H-L-Flanke am CLK/TRG-Anschluß wird wirksam	Ø
5	Der Vorteiler teilt den System-Takt durch 256	1
6	Der CTC arbeitet im *Timer Mode*	Ø
7	Interrupt sperren	Ø

Das Bild Ü9.1a zeigt das Bitmuster des *Channel Control Words.* In der Sedezimal-Darstellung ergibt sich für das *Channel Control Word* das Byte 2D.

b) Die Bits für das *Channel Control Word* müssen folgende Werte haben:

Bit	Programmierung	Wert
Ø	Kennung des *Channel Control Words*	1
1	Software-Reset	1
2	Es folgt ein *Time Constant Word*	1
3	Ohne Bedeutung: *Don't Care*, beliebiger Wert	1
4	Die L-H-Flanke am CLK/TRG-Anschluß wird wirksam	1
5	Ohne Bedeutung: *Don't Care*, beliebiger Wert	1
6	Der CTC arbeitet im *Counter Mode*	1
7	Interrupt freigeben	1

Das Bild Ü9.1b zeigt das Bitmuster des *Channel Control Words.* In der Sedezimal-Darstellung ergibt sich für das *Channel Control Word* das Byte ØFFH.

Die Aufgabenstellung verlangt genau das *Channel Control Word,* das zur Programmierung des CTC im Ereignis-Zähler-Programm (Seite L45) notwendig ist. Wir haben bei der hier angegebenen Lösung lediglich als *Don't Care*-Werte jeweils 1 eingesetzt.

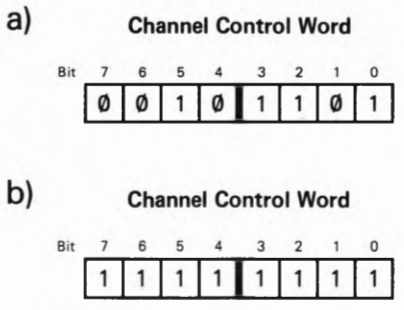


Bild Ü9.1
Zu Aufgabe S52.1: Bitmuster der Channel Control Words bei den Lösungen der Teilaufgaben a) und b).

Zu Aufgabe S54.1

Bei einer 50-Hz-Wechselspannung beträgt die Periodendauer 1/50 s = 0,02 s = 20 ms. Die Aufstellung der programmierbaren Zeiten für das Leerzählen des Abwärts-Zählers im CTC auf der Seite S54 zeigt, daß 20 ms bei einer Teilung durch 256 im Vorteiler erreichbar sind. Für das *Time Constant Word* ergibt sich demnach

TCW = 20 x 6,975 = 139,5 140 = 8CH

Weil nicht mit Interrupts gearbeitet werden soll, erübrigt sich die Programmierung eines *Interrupt Vector Words.*

Zum besseren Verständnis geben wir Ihnen hier die Programmierung mit Kommentaren an:

```
INIT:  ;Unterprogramm: Der CTC wird für die Ausgabe einer 50-Hz-Impuls-
       ;-------------- Spannung ohne Interrupt programmiert
       ;     CTC Kanal Ø
       ;                    Channel Control Word
       LD     A,ØØ1Ø1111B    ;Bit Nr.Ø: Kennung
                             ;Bit Nr.1: Reset
                             ;Bit Nr.2: Es folgt Zeitkonstante
                             ;Bit Nr.3: Triggern von CLK/TRG1
                             ;Bit Nr.4: mit 1-Ø-Übergang
                             ;Bit Nr.5: Vorteiler teilt durch 256
                             ;Bit Nr.6: Timer Mode
                             ;Bit Nr.7: Interrupt sperren
       OUT    (CTC1),A
       ;                    Time Constant Word
       LD     A,8CH          ;2Ø ms
       OUT    (CTC1),A
       RET                   ;Ende der Routine
```

Vergleichen Sie die Codierung im *Channel Control Word* bitte mit der Darstellung im Bild S 50.1!

Zu Aufgabe S 57.1

a) Die erste Adresse des Vektorfelds muß in der Sedezimal-Darstellung mit der Ziffer Ø oder mit der Ziffer 8 abschließen.

Der Befehl JP ist ein Drei-Byte-Befehl. Das Bild Ü 10.1 zeigt, daß im gewählten Beispiel der zweite Operand des Befehls bei der Adresse 187ØH steht. Bei dieser Adresse kann also das Vektorfeld zur Ablage der Anfangs-Adressen von Interrupt-Service-Routinen nicht beginnen.

Die nächstmögliche Adresse für den Beginn des Vektorfelds ist 1878H. Der CTC muß also mit dem *Interrupt Vector Word* 78H programmiert werden.

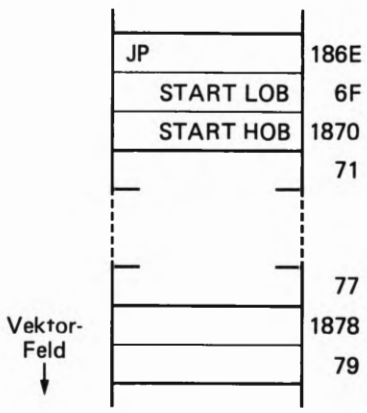


Bild Ü 10.1
Zu Aufgabe S 57.1: Wenn die Speicherzelle mit der Adresse 187ØH von Programmdaten belegt ist, dann muß das Interrupt Vektor Word für den Kanal Ø den Wert 78H haben.

b) Wenn der CTC mit dem *Interrupt Vector Word* 7ØH programmiert wird, dann beginnt das Vektorfeld bei der Adresse 187ØH. Ein vom Kanal Ø gelieferter Interrupt-Vektor würde auf diese – vom Programm belegte – Adresse weisen.

Wenn der Kanal Ø vom Programm nicht verwendet wird, dann schickt der CTC auch seinen Interrupt-Vektor nicht an die CPU. Es kann also auch keine falsche Adresse für eine Interrupt-Service-Routine abgeholt werden.

Bei Verwendung des Kanals 1 schickt dieser nach einer Interrupt-Anforderung den Interrupt-Vektor 72H mit dem Bitmuster ØØ11 ØØ1Ø an die CPU. Weil die Adresse 1872H nicht mit Programm-Daten belegt ist, kann dort die Anfangs-Adresse der Interrupt-Service-Routine für den Kanal 1 abgelegt werden.

Lösungen der im Text gestellten Aufgaben

Bitte sehen Sie sich die folgenden Lösungen erst dann an, wenn Sie die im Text gestellten Aufgaben selbständig durchgearbeitet haben.

Zu Aufgabe H66.1

Wenn die in Bild H65.1 dargestellten Adreßbits A1 und AØ an die Adreßleitungen A5 und A4 angeschlossen werden, ist zunächst einmal die niederwertige Hälfte des Steuerworts bedeutungslos. Wenn Sie für diese 4 Bits den Wert 1 einsetzen, ergibt sich folgendes:

Adresse:	Binär:	Sedezimal:
PORT A	11ØØ 1111	CF
PORT B	11Ø1 1111	DF
PORT C	111Ø 1111	EF
STEUERW.REG.	1111 1111	FF

Zu Aufgabe H70.1

Die Sedezimalzahlen lauten von oben nach unten gelesen:

8Ø, 81, 82, 83, 88, 89, 8A, 8B, 9Ø, 91, 92, 93, 98, 99, 9A, 9B.

Zu Aufgabe S72.1

Wenn der Inkrementier-Vorgang für das Register L weggelassen wird, arbeitet das Programm scheinbar immer noch gleich. Das Unterprogramm Zeit wird allerdings einmal weniger aufgerufen als die an Port C eingelesene Zahl es eigentlich erfordert.

Zu Aufgabe S74.1

```
1800  3E 90       LD    A,10010000B
1802  D3 03       OUT   (03H),A
1804  3E 01       LD    A,00000001B
1806  D3 02       OUT   (02H),A
1818  3E 30       LD    A,00110000B
181A  D3 01       OUT   (01H),A

181D  76          HALT
```

Zu Aufgabe S 81.1

Die Codierungen für die 7-Segment-Anzeige lauten:

Dezimalzahl	7-Segment-Code
1	3Ø
2	9B
3	BA
4	36
5	AE
6	AF

Zu Aufgabe S 89.1

Die Binärstruktur des Steuerworts für die Betriebsart 1; Port A und Port B arbeiten als Eingabeports; die an Port C verbleibenden Bits als Ausgabebits lauten:

1Ø11 Ø11X; wenn die mit X gekennzeichnete Stelle der Bitstruktur mit einer 1 belegt wird, ergibt sich die Sedezimalzahl B7H.

Zu Aufgabe S 93.1

Die Bitstruktur für das gesuchte Steuerwort lautet:

Ø111 11Ø1. = 7DH

Lehrgang PERIPHERIE-BAUSTEINE Verfasser: Hans Fischer, Edgar Hoch
Herausgeber: R. Christiani

Tabellen

7-Bit-Code ASCII

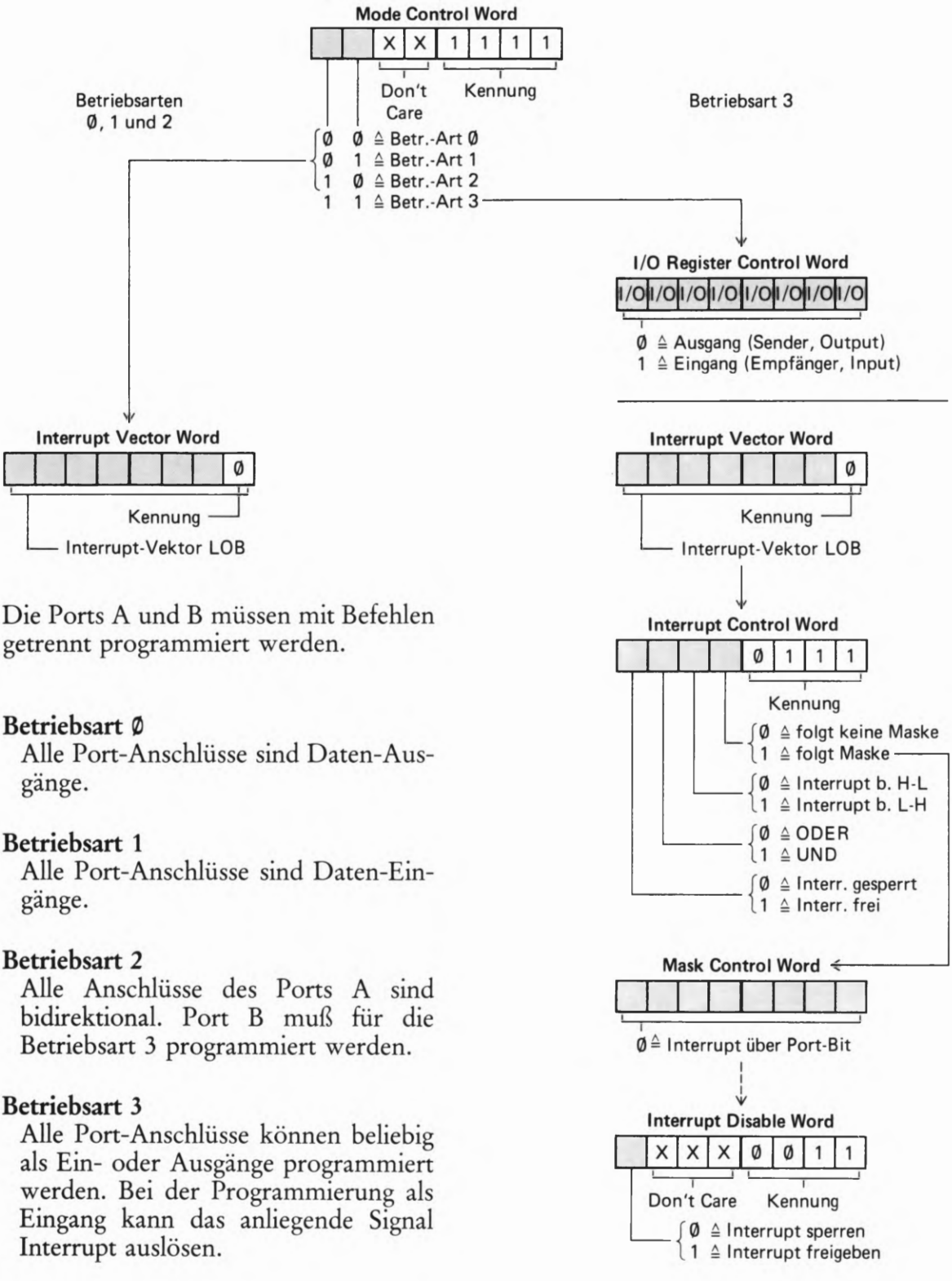
Steuerzeichen		Schriftzeichen		Schriftzeichen		Schriftzeichen	
Zeichen	Sedez.	Zeichen	Sedez.	Zeichen	Sedez.	Zeichen	Sedez.
NUL	ØØ	SP	2Ø	@ §	4Ø		6Ø
SOH	Ø1	!	21	A	41	a	61
STX	Ø2	"	22	B	42	b	62
ETX	Ø3	#	23	C	43	c	63
EOT	Ø4	$	24	D	44	d	64
ENQ	Ø5	%	25	E	45	e	65
ACK	Ø6	&	26	F	46	f	66
BEL	Ø7	´	27	G	47	g	67
BS	Ø8	(	28	H	48	h	68
HT	Ø9	)	29	I	49	i	69
LF	ØA	*	2A	J	4A	j	6A
VT	ØB	+	2B	K	4B	k	6B
FF	ØC	,	2C	L	4C	l	6C
CR	ØD	-	2D	M	4D	m	6D
SO	ØE	.	2E	N	4E	n	6E
SI	ØF	/	2F	O	4F	o	6F
DLE	1Ø	Ø	3Ø	P	5Ø	p	7Ø
DC1	11	1	31	Q	51	q	71
DC2	12	2	32	R	52	r	72
DC3	13	3	33	S	53	s	73
DC4	14	4	34	T	54	t	74
NAK	15	5	35	U	55	u	75
SYN	16	6	36	V	56	v	76
ETB	17	7	37	W	57	w	77
CAN	18	8	38	X	58	x	78
EM	19	9	39	Y	59	y	79
SUB	1A	:	3A	Z	5A	z	7A
ESC	1B	;	3B	[Ä	5B	{ä	7B
FS	1C	<	3C	\Ö	5C	| ö	7C
GS	1D	=	3D	] Ü	5D	}ü	7D
RS	1E	>	3E	↑ ^	5E	~ ß	7E
US	1F	?	3F	← -	5F	DEL	7F

Die Bedeutung der Steuerzeichen wird auf der Seite T2 erläutert.

Steuerzeichen des 7-Bit-Codes ASCII

Steuerzeichen		Bedeutung	
Zeichen	Sedez.	Englisch	Deutsch
NUL	ØØ	Null	Null, Nichts
SOH	Ø1	Start of Heading	Kopfzeilen-Beginn
STX	Ø2	Start of Text	Zeichen für Text-Anfang
ETX	Ø3	End of Text	Zeichen für Text-Ende
EOT	Ø4	End of Transmission	Ende der Übertragung
ENQ	Ø5	Enquiry	Aufford. z. Datenübertragung
ACK	Ø6	Acknowledge	Positive Rückmeldung
BEL	Ø7	Bell	Klingel
BS	Ø8	Backspace	Rückwärts-Schritt
HT	Ø9	Horizontal Tabulation	Horizontal-Tabulator
LF	ØA	Line-Feed	Zeilen-Vorschub
VT	ØB	Vertical Tabulation	Vertikal-Tabulator
FF	ØC	Form Feed	Seiten-Vorschub
CR	ØD	Carriage Return	Wagen-Rücklauf
SO	ØE	Shift Out	Dauerumschalt-Zeichen
SI	ØF	Shift In	Rückschaltungs-Zeichen
DLE	1Ø	Data Link Escape	Umschalt. f. Daten-Übertrag.
DC1	11 XON	Device Control 1	Geräte-Steuerzeichen 1
DC2	12	Device Control 2	Geräte-Steuerzeichen 2
DC3	13 XOFF	Device Control 3	Geräte-Steuerzeichen 3
DC4	14	Device Control 4	Geräte-Steuerzeichen 4
NAK	15	Negative Acknowledge	Negative Rückmeldung
SYN	16	Synchronous Idle	Synchronisierung
ETB	17	End of Transmission Block	Ende Datenübertragungs-Block
CAN	18	Cancel	Ungültig
EM	19	End of Medium	Ende der Aufzeichnung
SUB	1A	Substitute	Ersatz
ESC	1B	Escape	Umschaltung
FS	1C	File Separator	Hauptgruppen-Trennung
GS	1D	Group Separator	Gruppen-Trennung
RS	1E	Record Separator	Untergruppen-Trennung
US	1F	Unit Separator	Teilegruppen-Trennung

Programmierung der Z 80 PIO



Die Ports A und B müssen mit Befehlen getrennt programmiert werden.

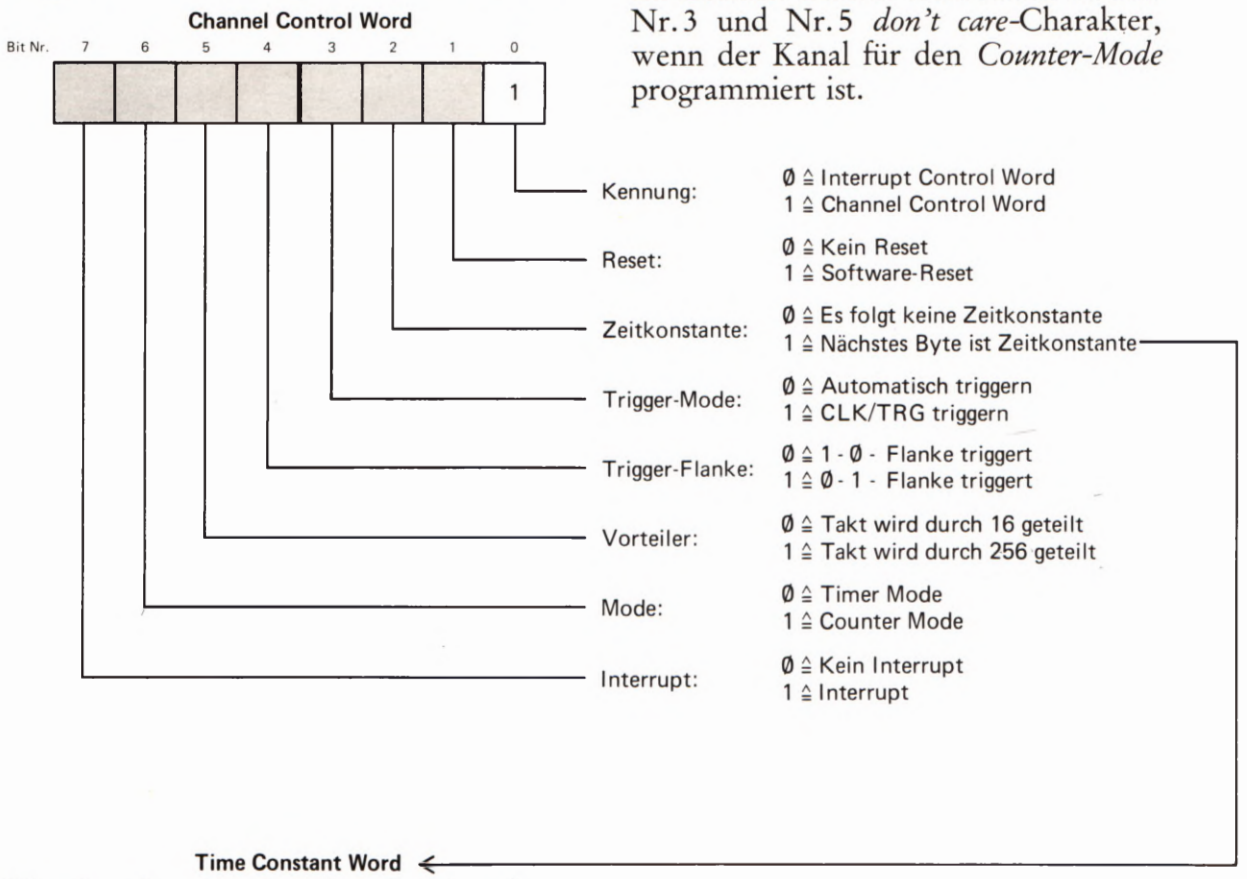
Betriebsart Ø
Alle Port-Anschlüsse sind Daten-Ausgänge.

Betriebsart 1
Alle Port-Anschlüsse sind Daten-Eingänge.

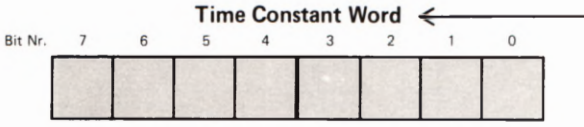
Betriebsart 2
Alle Anschlüsse des Ports A sind bidirektional. Port B muß für die Betriebsart 3 programmiert werden.

Betriebsart 3
Alle Port-Anschlüsse können beliebig als Ein- oder Ausgänge programmiert werden. Bei der Programmierung als Eingang kann das anliegende Signal Interrupt auslösen.

Programmierung des Z 80 CTC



Im *Channel Control Word* haben die Bits Nr. 3 und Nr. 5 *don't care*-Charakter, wenn der Kanal für den *Counter-Mode* programmiert ist.



Bei der Taktfrequenz 1,79 MHz des Micro-Professors wird das *Time Constant Word* TCW mit folgenden Formeln berechnet, wenn die Zeitkonstante Tv in ms eingesetzt wird:

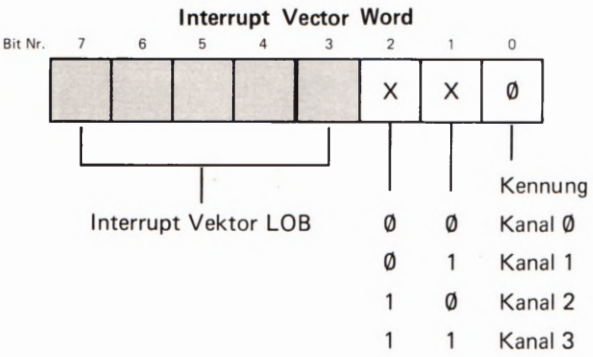
Vorteiler teilt durch 16:

TCW = Tv x 111,61

Vorteiler teilt durch 256:

TCW = Tv x 6,975

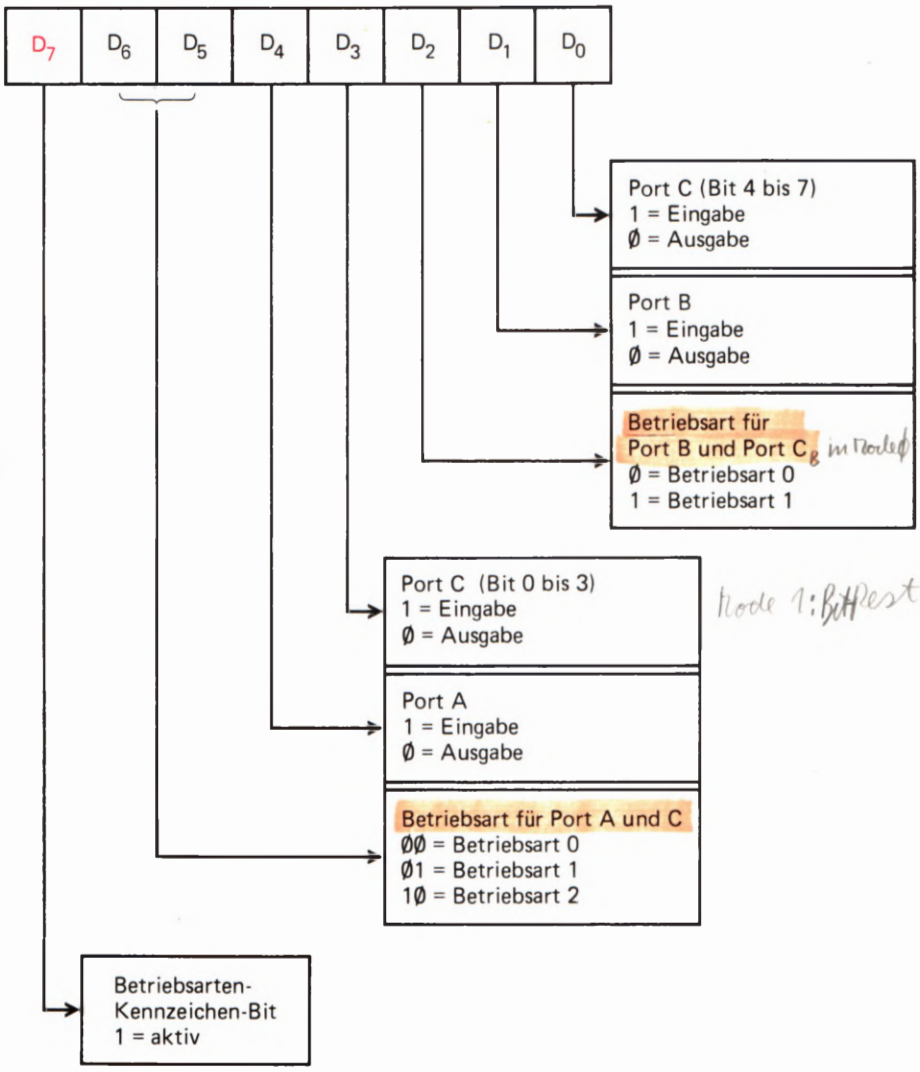
Der sich ergebende Wert muß in die Sedezimal-Darstellung umgewandelt werden.



Das *Interrupt Vector Word* braucht für alle vier Kanäle gemeinsam nur einmal programmiert zu werden. Die Bits Nr. 1 und 2 haben *don't care*-Charakter.

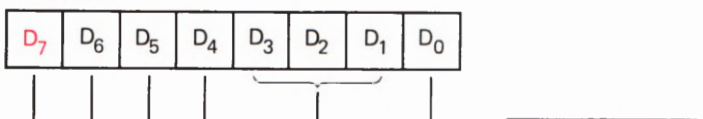
Programmierung des 8255

Bedeutung des Steuerworts



Setzen und Rücksetzen einzelner Bits am Port C

Steuerwort



Kennzeichenbit = Ø

Bit-Auswahl an Port C

	7	6	5	4	3	2	1	0	Bit-Nr.
D_1	1	Ø	1	Ø	1	Ø	1	Ø	
D_2	1	1	Ø	Ø	1	1	Ø	Ø	
D_3	1	1	1	1	Ø	Ø	Ø	Ø	

Lehrgang PERIPHERIE-BAUSTEINE Verfasser: Hans Fischer, Edgar Hoch
Herausgeber: R. Christiani

Programm-Listen

Das Assembler-Format

Auf den folgenden Seiten finden Sie die Auflistungen der Programme, die in den Versuchen im Lehrbrief verwendet werden. Die Form dieser Auflistungen weicht von der Ihnen bekannten Form im Lehrgang Mikroprozessortechnik ab. Es ist eine Form, wie sie bei der Verwendung eines Assembler-Programms benutzt wird.

Ein Assembler-Programm übersetzt in mnemonischen Codes geschriebene Befehle in die Bytes der Maschinensprache. (Vgl. Seite S25 im Lehrgang Mikroprozessortechnik.) Damit das richtig funktioniert, müssen beim Anschreiben des Programms in mnemonischen Codes bestimmte Regeln beachtet werden. Diese Regeln wollen wir Ihnen hier soweit erläutern, wie es zum Lesen und zum Verständnis der Programm-Auflistungen notwendig ist.

Ein flüchtiger Blick auf die Programm-Auflistungen zeigt Ihnen bereits, welches Ergebnis die Benutzung eines Assembler-Programms liefert. Auf den rechten zwei Dritteln einer jeden Seite steht das in mnemonischen Codes eingegebene Programm. Außerdem stehen dort Anmerkungen, die mit dem Programm selbst nichts zu tun haben, die aber das Verständnis des Programms erleichtern.

Im linken Drittel einer jeden Seite sehen Sie die zu den einzelnen Befehlen gehörenden Bytes, die das Assembler-Programm aus den mnemonischen Codes der Befehle generiert hat.

Ganz links finden Sie die Adresse des ersten zu einem Befehl gehörenden Bytes. Rechts daneben stehen dann die Bytes, die den Befehl in der Maschinensprache darstellen. Bei einem Ein-Byte-Befehl steht dort nur ein einziges Byte; bei Mehr-Byte-Befehlen stehen die Bytes nebeneinander. Dieser Teil der Auflistung entspricht genau dem rot unterlegten Teil der Programm-Auflistung im Bild H33.1 des Lehrgangs Mikroprozessortechnik.

Für die Übersetzungs-Aufgabe des Assembler-Programms – man sagt kurz: für den Assembler – sind nur die mnemonischen Codes der Befehle und deren Operanden interessant. Die erläuternden Anmerkungen könnten auch weggelassen werden. Der Assembler wird auch ohne diese Anmerkungen die Bytes der Maschinensprache liefern. Die Anmerkungen würden den Assembler sogar hoffnungslos verwirren, denn er versteht nur die mnemonischen Befehls-Codes. Man muß ihm deshalb eigens deutlich machen, um welchen Teil der ihm zur Übersetzung vorgelegten Programm-Liste er sich gefälligst nicht zu kümmern hat. (Man bezeichnet diese Programm-Liste in der Fachsprache als **Quellprogramm** oder auf englisch als *Source-Program.*)

Die Kennzeichnung von Anmerkungen wird im Quellprogramm durch einen Strichpunkt (Semikolon) vorgenommen. Alles, was in einer Zeile des Quellprogramms rechts von einem Strichpunkt steht, wird vom Assembler einfach nicht zur Kenntnis genommen.

Zu den besonderen Fähigkeiten eines Assemblers gehört es, daß er den Umgang mit Labels beherrscht. (Vgl. Lehrgang Mikroprozessortechnik, Seite H32.) Wenn also an irgendeiner Stelle des Quellprogramms ein Sprung- oder ein CALL-Befehl als Operanden einen Label hat, dann sucht der Assembler das Quellprogramm nach dem Befehl ab,

der mit diesem Label (Etikett) gekennzeichnet ist. Bei CALL-Befehlen wird dann anstelle des Labels im Operanden die Adresse des mit dem Label gekennzeichneten Befehls eingesetzt. Steht der Label als Operand eines Sprungsbefehls mit relativer Adressierung (vgl. Lehrgang Mikroprozessortechnik ab der Seite S78), dann rechnet der Assembler die Sprungweite zu dem mit dem Label gekennzeichneten Befehl aus und setzt in der Maschinensprache das entsprechende Byte ein.

Manchmal kommt es vor, daß in einem Befehl ein Label als Operand eingesetzt wird, der im Verlauf des Programms nirgendwo zur Kennzeichnung eines bestimmten Befehls verwendet wird. In einem solchen Fall ist die Bezeichnung Label (Etikett) also eigentlich gar nicht angebracht. Man spricht dann von einer symbolischen Adresse. Damit der Assembler bei der Übersetzung nicht vergeblich nach dem wirklichen Wert dieser symbolischen Adresse suchen muß, weist man der symbolischen Adresse am Anfang des Programms einen Wert zu. Das geschieht mit der Assembler-Anweisung EQU. EQU ist die Abkürzung von **EQU***als* und bedeutet: Ist gleich. Die entsprechende Anweisung heißt dann z. B.:

KIN EQU ØØH (KIN *EQUals* ØØH, KIN ist gleich ØØH)

Der Assembler setzt jetzt immer dort, wo er die symbolische Adresse KIN findet, das Byte ØØ ein.

Eine weitere Assembler-Anweisung werden Sie am Anfang aller Programme finden: Die ORG-Anweisung. Mit dieser Anweisung wird dem Assembler mitgeteilt, welche Adresse er dem ersten Befehl des Programms zuweisen soll. Bei unseren Programmen für den Micro-Professor wird das vermutlich immer die Adresse 18ØØH sein.

Besonders auffallen wird Ihnen, daß wir Ziffern im Quellprogramm meist den Buchstaben H folgen lassen. Der Grund dafür ist, daß der Assembler dezimal dargestellte Zahlen richtig interpretiert. Wenn der Assembler z. B. die Anweisung CALL 1845 findet, dann interpretiert er 1845 als Dezimaldarstellung der Zahl Eintausendachthundertfünfundvierzig und übersetzt das mit CD 35 Ø7. Er wandelt also die Dezimal-Darstellung 1845 in die Sedezimal-Darstellung Ø735 der Zahl Eintausendachthundertfünfundvierzig um.

Will man dem Assembler begreiflich machen, daß man mit 1845 bereits eine sedezimal dargestellte Adresse meint, dann muß man die Ziffernfolge 1845 durch den nachgestellten Buchstaben H als Sedezimal-Darstellung (**H**exadezimal-Darstellung) kennzeichnen.

Der Assembler versteht sogar die Binär-Darstellung von Zahlen, wenn man eine Ziffernfolge mit dem nachgestellten Buchstaben B als **B**inär-Darstellung kennzeichnet.

Und noch eine Besonderheit: Durch Ziffern dargestellte Zahlen müssen immer mit einer der Ziffern 0 bis 9 beginnen. – In der Sedezimal-Darstellung sind zwar die Zeichen A bis F ebenfalls Ziffern, aber das erkennt der Assembler nicht ohne weiteres. Der Assembler wird z. B. im Befehl CALL BADE den Operanden BADE nicht als sedezimal dargestellte Adresse interpretieren, sondern als Label BADE eines Befehls im Programm. Wenn mit BADE eine sedezimal dargestellte Adresse gemeint ist, dann muß im Quellprogramm der Befehl CALL ØBADEH lauten.

```
                        ;************************************************
                        ;*                                              *
                        ;*       Serielle Ausgabe von Daten-Bytes       *
                        ;*     mit optischer und akustischer  Anzeige   *
                        ;*                                              *
                        ;************************************************

                        ;Anfangs-Adresse des auszugebenden Bereichs:

                        ANFAD   EQU     1800H

                        ;Definitionen:

                        KIN     EQU     00H     ;8255 Port A
                        SEG7    EQU     01H     ;8255 Port B
                        DIGIT   EQU     02H     ;8255 Port C

                                ORG     1800H

                        ;************************************************

                        START:  ;Anfangsbedingungen

1800  CD EF 18                    CALL    INIT    ;Initialisierung

                        ;************************************************

                        BEREIT: ;Hauptprogramm
                                ;-------------
                                ;Nach Empfang eines CTS- (Clear-To-Send-)
                                ;Signals wird das jeweils folgende Byte im
                                ;adressierten Bereich seriell ausgegeben.
                                ;Die Ausgabe eines Bytes beginnt mit einem
                                ;Start-0-Bit. Es folgen die 8 Daten-Bits.
                                ;Die Ausgabe eines Bytes wird mit 2 Stop-
                                ;1-Bits abgeschlossen.

1803  CD 0D 18                    CALL    CTS     ;Warten auf CTS-Signal
1806  7E                          LD      A,(HL)  ;Byte holen
1807  CD 19 18                    CALL    SEROUT  ;    und seriell ausgeben
180A  23                          INC     HL      ;Zeiger auf nächstes Byte
180B  18 F6                       JR      BEREIT  ;Nächstes Byte

                        ;************************************************
```

```
                        ;************************************************

                        CTS:    ;Unterprogramm: Clear To Send
                                ;-------------
                                ;Das Programm wartet in einer Schleife
                                ;auf das Clear-To-Send-Signal von der Pe-
                                ;ripherie.
                                ;Im Micro-Professor wird das CTS-Signal
                                ;mit der User-Key simuliert.
                                ;In der Schleife wird die 7-Segment-Anzei-
                                ;ge mit der DISPL-Routine bedient.

180D  E5                        PUSH    HL

180E  CD D0 18                  CALL    DISPL       ;Anzeige bedienen
1811  DB 00                    IN      A,(KIN)     ;Taste abfragen und
1813  E6 40                    AND     01000000B   ;    maskieren
1815  20 F7                    JR      NZ,CTS+1    ;Kein CTS-Signal
1857  E1                        POP     HL
1818  C9                        RET

                        ;************************************************

                        SEROUT: ;Unterprogramm: Serielle Byte-Ausgabe
                                ;-------------
                                ;Zuerst wird ein Start-0-Bit ausgegeben,
                                ;anschließend die 8 Daten-Bits. Die Aus-
                                ;gabe wird mit 2 Stop-1-Bits abgeschlos-
                                ;sen. Anschließend wird der Anzeige-Poin-
                                ;ter zurück (nach rechts) gesetzt.
                                ;Das im Akkumulator übergebene Byte wird
                                ;im C-Register abgelegt und von dort Bit
                                ;für Bit in das Carry-Flag geschoben.

1819  E5                        PUSH    HL

181A  4F                        LD      C,A     ;Byte nach C
181B  06 08                     LD      B,8     ;Zähler für 8 Bits

181D  CD 38 18                  CALL    STARTB  ;Ausgabe Start-0-Bit

1820  CB 19                BIT:  RR      C       ;Bit aus C nach CY
1822  DC 4A 18                  CALL    C,HIAUS ;1-Bit ausgeben
1825  D4 59 18                  CALL    NC,LOAUS ;0-Bit ausgeben
1828  10 F6                     DJNZ    BIT     ;Insgesamt 8 Bit

182A  CD 75 18                  CALL    STOPB   ;2 Stop-Bits ausgeben
182D  CD 75 18                  CALL    STOPB

                                ;Anzeige  Bitmusters im Versuch vorbereiten

1830  21 0E 19                  LD      HL,DSTACK       ;Pointer rechts
1833  22 0C 19                  LD      (LED),HL        ;       ablegen

1836  E1                        POP     HL
1837  C9                        RET
```

```
                        ;************************************************

                                ;Die Ausgabe der Bits erfolgt im Versuch
                                ;optisch über die 7-Segment-Anzeige und
                                ;akustisch während der WARTE-Zeit über
                                ;den Lautsprecher.
                                ;Für die optische Anzeige werden die Bits
                                ;als unterschiedliche 7-Segment-Muster in
                                ;einem Anzeige-Buffer abgelegt.

                        STARTB: ;Unterprogramm: Ausgabe und Anzeige des
                                ;------------  Start-0-Bits

1838  C5                        PUSH    BC
1839  E5                        PUSH    HL

183A  CD C1 18                  CALL    LOESCH          ;Anzeige löschen

183D  2A 03 19                  LD      HL.(STATON)     ;Parameter für
1840  3A 05 19                  LD      A.(STATON+2)    ;    die Ton-Höhe
1843  4F                        LD      C.A

1844  CD 97 18                  CALL    WARTE           ;1 Bit-Zeit

1847  E1                        POP     HL
1848  C1                        POP     BC
1849  C9                        RET

                        ;************************************************

                        HIAUS:  ;Unterprogramm: Ausgabe und Anzeige eines
                                ;------------  1-Bits.

184A  3E 08                     LD      A.00001000B     ;7-Segm.-Bitmuster
184C  00 00 00                  NOP! NOP! NOP           ;    ausgeben

184F  3A 0E 19                  LD      A.(DSTACK)      ;Bit auf rechtem
1852  F6 40                     OR      01000000B       ;    Dezimalpunkt
1854  32 0E 19                  LD      (DSTACK).A      ;        ausgeben

1857  18 0D                     JR      DELAY

                        ;************************************************

                        LOAUS:  ;Unterprogramm: Ausgabe und Anzeige eines
                                ;------------  0-Bits

1859  3E 02                     LD      A.00000010B     ;7-Segm.-Bitmuster
185B  00 00 00                  NOP! NOP! NOP           ;    ausgeben

185E  3A 0E 19                  LD      A.(DSTACK)      ;Bit auf rechtem
1861  E6 BF                     AND     10111111B       ;    Dezimalpunkt
1863  32 0E 19                  LD      (DSTACK).A      ;        ausgeben

                        ;************************************************
```

```
                        DELAY:  ;Unterprogramm: Verzögerung mit Ton-
                                ;------------  Ausgabe

1866  C5                        PUSH    BC
1867  E5                        PUSH    HL

1868  2A 06 19                  LD      HL.(BITTON)     ;Parameter für
186B  3A 08 19                  LD      A.(BITTON+2)    ;    die Ton-Höhe
186E  4F                        LD      C.A

186F  CD 97 18                  CALL    WARTE           ;1 Bit-Zeit

1872  E1                        POP     HL
1873  C1                        POP     BC
1874  C9                        RET

                        ;***********************************************

                        STOPB:  ;Unterprogramm: Stop-Bit ausgeben
                                ;------------

1875  3A 0E 19                  LD      A.(DSTACK)      ;Dezimal-Punkt in
1878  F6 40                     OR      01000000B       ;    der Anzeige
187A  32 0E 19                  LD      (DSTACK).A      ;          setzen

187D  C5                        PUSH    BC
187E  E5                        PUSH    HL

187F  2A 09 19                  LD      HL.(STTON)      ;Parameter für
1882  3A 0B 19                  LD      A.(STTON+2)     ;    die Ton-Höhe
1885  4F                        LD      C.A

1886  CD 97 18                  CALL    WARTE           ;1 Bit-Zeit

1889  E1                        POP     HL
188A  C1                        POP     BC
188B  C9                        RET

                        ;***********************************************

                        AUSGAB: ;Unterprogramm: Bitmuster-Anzeige
                                ;------------
                                ;Die seriell ausgegebenen Bits werden als
                                ;7-Segment-Muster in aufeinanderfolgenden
                                ;Speicherzellen des Anzeige-Buffers abge-
                                ;legt. Pointer Anz.-Buffer inkrementieren.

188C  E5                        PUSH    HL

188D  2A 0C 19                  LD      HL.(LED)        ;Stelle holen
1890  77                        LD      (HL).A          ;Muster nach Buffer
1891  23                        INC     HL              ;Nächste Stelle
1892  22 0C 19                  LD      (LED).HL        ;          ablegen

1895  E1                        POP     HL
1896  C9                        RET
```

```
                        ;************************************************

                        WARTE:  ;Unterprogramm: Warte-Schleife
                                ;-------------
                                ;Nach der Ausgabe eines Bits läuft das Pro-
                                ;gramm in zwei aufeinanderfolgenden. gleich
                                ;langen Warteschleifen. deren Dauer die Aus-
                                ;gabe-Geschwindigkeit (Baud-Rate) bestimmt.

1897  CD 9E 18                  CALL    TON             ;Warten mit Ton
189A  CD B3 18                  CALL    HALBBIT         ;Warten ohne Ton
189D  C9                        RET

                        ;************************************************

                        TON:    ;Unterprogramm: Ton-Ausgabe
                                ;-------------
                                ;Perioden-Dauer und Anzahl werden dem Un-
                                ;terprogramm als Parameter in den HL- und
                                ;C-Registern übergeben.
                                ;Während des Tons wird die 7-Segment-Anzei-
                                ;ge mit der DISPL-Routine bedient.

189E  29                        ADD     HL.HL           ;Halb-Perioden!
189F  11 01 00                  LD      DE.1
18A2  3E FF                     LD      A.11111111B

18A4  D3 02             SQWAVE: OUT     (DIGIT).A       ;Ausgabe an Port
18A6  41                        LD      B.C
18A7  CD D0 18          ANZ:    CALL    DISPL           ;Anzeige bedienen
18AA  10 FB                     DJNZ    ANZ
18AC  EE 80                     XOR     80H             ;Ausgabe wechseln
18AE  ED 52                     SBC     HL.DE
18B0  20 F2                     JR      NZ.SQWAVE

18B2  C9                        RET

                        ;************************************************

                        HALBBIT:;Unterprogramm: ;Warte-Schleife ohne Ton
                                ;-------------
                                ;In der Schleife wird die 7-Segment-Anzei-
                                ;ge mit der DISPL-Routine bedient.

18B3  21 26 F0                  LD      HL.0F026H
18B6  11 01 00                  LD      DE.1

18B9  CD D0 18          LOOP:   CALL    DISPL           ;Anzeige bedienen
18BC  ED 5A                     ADC     HL.DE
18BE  30 F9                     JR      NC.LOOP

18C0  C9                        RET

                        ;************************************************
```

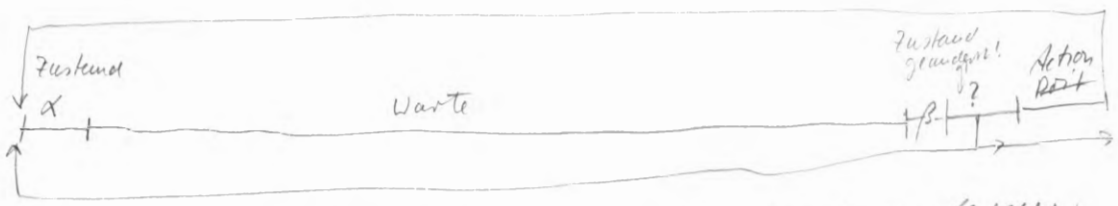
```
                        ;************************************************

                        LOESCH: ;Unterprogramm: Anzeige löschen
                                ;------------
                                ;Die Anzeige wird durch Löschen des Anzei-
                                ;ge-Buffers abgeschaltet.

18C1  C5                        PUSH    BC
18C2  E5                        PUSH    HL

18C3  06 06                     LD      B.6             ;6 Anzeige-Stellen
18C5  21 13 19                  LD      HL.DSTACK+5     ;Pointer auf Buffer

18C8  36 00             NEXT:   LD      (HL).0
18CA  2B                        DEC     HL
18CB  10 FB                     DJNZ    NEXT

18CD  E1                        POP     HL
18CE  C1                        POP     BC
18CF  C9                        RET

                        ;************************************************

                        DISPL:  ;Unterprogramm: Bit anzeigen
                                ;------------
                                ;Bei jedem Aufruf dieser Routine wird der
                                ;Inhalt der jeweils nächsten Stelle im
                                ;Anzeige-Buffer in die entsprechende Stel-
                                ;le der 7-Segment-Anzeige geschrieben.

18D0  08                        EX      AF.AF'          ;2. Registersatz
18D1  D9                        EXX

18D2  AF                        XOR     A               ;(A) = 0
18D3  B8                        CP      B               ;(Zähler) = 0?
18D4  20 07                     JR      NZ.STELLE       ;  Nein!

18D6  06 06                     LD      B.6             ;Wieder von vorn
18D8  1E C1                     LD      E.11000001B     ; rechte Stelle
18DA  21 0E 19                  LD      HL.DSTACK

18DD  ED 79             STELLE: OUT     (C).A           ;Muster löschen
18DF  7B                        LD      A.E
18E0  D3 02                     OUT     (DIGIT).A

18E2  ED A3                     OUTI                    ;Neues Muster
18E4  7B                        LD      A.E             ;Anzeigestelle
18E5  D3 02                     OUT     (DIGIT).A       ;     aktivieren

18E7  CB 27                     SLA     A               ;Nächste Stelle
18E9  F6 C0                     OR      11000000B
18EB  5F                        LD      E.A

18EC  D9                        EXX                     ;1. Registersatz
18ED  08                        EX      AF.AF'
18EE  C9                        RET
```

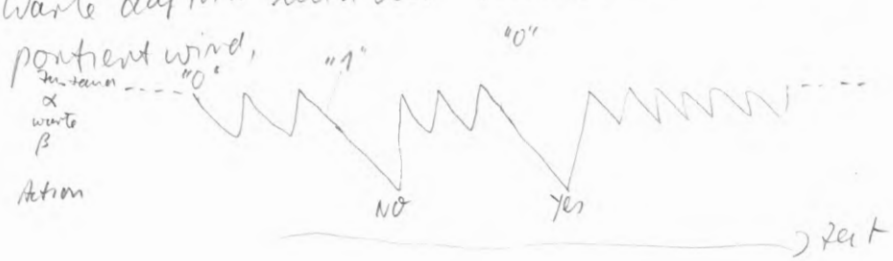
```
                        ;************************************************

                        INIT:   ;Unterprogramm: Anfangs-Bedingungen
                                ;-------------
                                ;Im 2. Register-Satz wird der Zähler B'
                                ;fÜr 6 Anzeige-Stellen gelöscht. In C'
                                ;wird die Port-Adresse für die 7-Segment-
                                ;Muster abgelegt. Anzeige-Buffer löschen.
                                ;Der Anzeige-Pointer wird auf die rechte
                                ;Anzeige-Stelle gesetzt. Dabei wird der
                                ;rechte Dezimal-Punkt zur Anzeige des Ru-
                                ;he-1-Signals eingeschaltet. Der Pointer
                                ;wird abgelegt. ;Die Anfangs-Adresse des
                                ;auszugebenden Bereichs wird geladen.

18EF  D9                        EXX                     ;2. Register-Satz
18F0  01 01 00                  LD      BC.SEG7
18F3  D9                        EXX                     ;1. Register-Satz

18F4  CD C1 18                  CALL    LOESCH          ;Anzeige löschen

18F7  21 0E 19                  LD      HL.DSTACK       ;Pointer ablegen
18FA  36 40                     LD      (HL).01000000B  ;Ruhe-1-Signal
18FC  22 0C 19                  LD      (LED).HL

18FF  21 00 18                  LD      HL.ANFAD        ;Anfang Bereich

1902  C9                        RET

                        ;************************************************

                                ;Ton-Parameter:
                                ;-------------
                                ;Die ersten beiden Bytes bestimmen jeweils
                                ;die Anzahl der Tonfrequenz-Perioden. Das
                                ;dritte Byte bestimmt die Perioden-Dauer.

1903  8C 00                  STATON: DW      008CH   ;Ton beim Start-Bit
1905  10                            DB      10H

1906  18 01                  BITTON: DW      0118H   ;Ton bei den Daten-Bits
1908  08                            DB      08H

1909  5E 01                  STTON:  DW      015EH   ;Ton bei den Stop-Bits
190B  06                            DB      06H

                        ;************************************************

                                ;Anzeige-Buffer
                                ;--------------

190C  00 00                     LED:    DS      2.0     ;Ablage f. Buffer-Pointer
190E  00 00 00 00 00 00         DSTACK: DS      6.0
                                zzzzzz: EQU     $-1
                        ;************************************************
                                END
```

```
                ;*************************************************
                ;*                                               *
                ;*   Die Ausgänge des PIO-Ports B werden Bit-    *
                ;*   weise über die Tasten des PIO-Ports A ein-  *
                ;*   und ausgeschaltet. (Flipflop-Schaltung.)    *
                ;*                                               *
                ;*************************************************

                ;Definitionen:

                PIODA   EQU     80H     ;Daten.   Port A
                PIODB   EQU     81H     ;Daten.   Port B
                PIOBA   EQU     82H     ;Befehle. Port A
                PIOBB   EQU     83H     ;Befehle. Port B

                        ORG     1800H

                ;*************************************************

                START:  ;Anfangs-Bedingungen

1800  CD 1A 18            CALL    INIT        ;PIO initialisieren

1803  3E 00               LD      A,0         ;Ausgänge löschen
1805  D3 81               OUT     (PIODB),A
1807  67                  LD      H,A         ;Muster in H ablegen

                ;*************************************************

                HOLE:   ;Hauptprogramm: Abfrage der Port-A-Tasten
                        ;------------  Umschalten der Port-B-Bits

1808  DB 80               IN      A,(PIODA)   ;Eingabe von Port A
180A  4F                  LD      C,A         ;  im C-Reg. verwahren
180B  CD 27 18            CALL    WARTE       ;Einen Augenblick später
180E  DB 80               IN      A,(PIODA)   ;  nochmal nachsehen
1810  B9                  CP      C           ;Solange unverändert.
1811  28 F5               JR      Z,HOLE      ;  weiter abfragen.

1813  79                  LD      A,C         ;Eingabe-Bit holen und
1814  AC                  XOR     H           ;  dieses Bit im H-Reg.
1815  67                  LD      H,A         ;  umschalten.
1816  D3 81               OUT     (PIODB),A   ;Neues Muster ausgeben

1818  18 EE               JR      HOLE        ;Neue Eingabe erwarten

                ;*************************************************
```

```
                ;************************************************

INIT:           ;Unterprogramm: Befehle für PIO
                ;-------------    Ports A und B

                        ;Port A:

181A  3E FF             LD      A,0FFH          ;Betriebsart 3
181C  D3 82             OUT     (PIOBA).A
181E  D3 82             OUT     (PIOBA).A       ;Bits Nr.7 - 0 EIN

                        ;Port B:

1820  D3 83             OUT     (PIOBB).A       ;Betriebsart 3
1822  3E 00             LD      A,00            ;Bits Nr.7 - 0 AUS
1824  D3 83             OUT     (PIOBB).A
1826  C9                RET

                ;************************************************

WARTE:          ;Unterprogramm: Verzögerung
                ;-------------

1827  06 FF             LD      B,0FFH
1829  10 FE    LOOP:     DJNZ    LOOP
182B  C9                RET

                ;************************************************

                        END
```

Zeitablauf





```
                        ;*********************************************
                        ;* Auf der Peripherie-Leiterplatte wird ein  *
                        ;* Leuchtpunkt in den beiden Leuchtdioden-Zei- *
                        ;* len innerhalb von 6 Stellen wahlweise     *
                        ;* rechts- und linksherum im Kreis geschoben. *
                        ;* Beim Z 80 PIO-Port B wird mit der Taste 7 *
                        ;* die Links-Richtung eingestellt, mit der   *
                        ;* Taste 0 die Rechts-Richtung.              *
                        ;* Beim Port A wird mit Taste 0 das Schieben *
                        ;* gestartet, mit Taste 7 angehalten.        *
                        ;*********************************************

                        ;Definitionen:

PIODA   EQU     80H             ;Daten,  Port A
PIODB   EQU     81H             ;Daten,  Port B
PIOBA   EQU     82H             ;Befehle. Port A
PIOBB   EQU     83H             ;Befehle. Port B

        ORG     1800H

;*****************************************************

START:  ;Anfangs-Bedingungen

1800  CD 1F 18    CALL    INIT            ;PIO initialisieren

1803  26 00       LD      H.00000000B     ;Anfangs-Muster
1805  2E 80       LD      L.10000000B

1807  0E 01       LD      C.00000001B     ;RECHTS. STOP

;*************************************************

RUND:   ;Hauptprogramm: Abfrage der Steuertasten
        ;------------  Umlauf-Geschwindigkeit
        ;              Steuerung der Leuchtdioden

1809  CD 3C 18    CALL    ABFRAG          ;Steuer-Tasten
180C  CD 30 18    CALL    WAIT            ;Geschwindigkeit

        ;              Die Steuer-Flags stehen in C:
        ;              Bit Nr.0: 0 = Links-Lauf
        ;                        1 = Rechts-Lauf
        ;              Bit Nr.1: 0 = Stop
        ;                        1 = Lauf

180F  79          LD      A.C
1810  E6 02       AND     00000010B       ;Stop oder Lauf?
1812  28 F5       JR      Z.RUND          ;  Bei 0 Stop
1814  79          LD      A.C
1815  E6 01       AND     00000001B       ;Rechts oder Links?
1817  C4 5F 18    CALL    NZ.RECHTS       ;  Bei 1 > Schieben
181A  CC 8F 18    CALL    Z.LINKS         ;  Bei 0 < Schieben

181D  18 EA       JR      RUND            ;Nächster Stellung
```

```
                        ;*****************************************************

                INIT:   ;Unterprogramm: Befehle für PIO Ports A und B
                        ;-------------
                                        ;Port A:

181F  3E FF             LD      A,0FFH          ;Betriebsart 3
1821  D3 82             OUT     (PIOBA),A
1823  3E 81             LD      A,81H           ;Bits Nr.7 u. Nr.1 EIN
1825  D3 82             OUT     (PIOBA),A       ;Bits Nr.6 -  Nr.2 AUS

                                        ;Port B:

1827  3E FF             LD      A,0FFH          ;Betriebsart 3
1829  D3 83             OUT     (PIOBB),A
182B  3E 81             LD      A,81H           ;Bits Nr.7 u. Nr.1 EIN
182D  D3 83             OUT     (PIOBB),A       ;Bits Nr.6 -  Nr.2 AUS

182F  C9                RET                     ;Ende der Routine

                        ;*****************************************************

                WAIT:   ;Unterprogramm: Verzögerung vor dem Schieben
                        ;-------------

1830  E5                PUSH    HL              ;(HL) retten

1831  21 00 E8          LD      HL,0E800H       ;(HL) mit (DE)=1 hoch-
1834  11 01 00          LD      DE,1            ;  zählen

1837  19          LOOP: ADD     HL,DE           ;(HL)+(DE) bis
1838  30 FD             JR      NC,LOOP         ;  (CY)=1, (HL)=0000 H

183A  E1                POP     HL              ;Alter Wert von (HL)
183B  C9                RET                     ;Ende der Routine

                        ;*****************************************************

                ABFRAG: ;Unterprogramm: Steuertasten abfragen
                        ;-------------  Flag im C-Register ablegen

                        ;               Links oder rechts

183C  DB 81             IN      A,(PIODB)       ;Port B
183E  E6 81             AND     10000001B       ;  Tasten Nr.7 u. Nr.0
1840  28 0C             JR      Z,LAUSTO        ;  Nicht betätigt

1842  E6 01             AND     00000001B       ;Taste Nr.0 ?
1844  20 06             JR      NZ,LARE         ;  Ja:   Lauf rechts
1846  3E 02             LD      A,00000010B     ;  Nein: Lauf links:
1848  A1                AND     C               ;         Bit Nr.0 = 0
1849  4F                LD      C,A             ;         im C-Reg.
184A  18 02             JR      LAUSTO          ;Lauf/Stop abfragen

184C  B1          LARE: OR      C               ;Lauf rechts
184D  4F                LD      C,A
```

ORA tuts hier nicht

```
                                ;              Lauf oder stop

184E  DB 80           LAUSTO: IN      A.(PIODA)       ;Port A
1850  E6 81                   AND     10000001B       ;  Tasten Nr.7 u. Nr.0
1852  C8                     RET     Z               ;  Nicht betätigt

1853  E6 80                   AND     10000000B       ;Taste Nr.7 ?
1855  28 03                   JR      Z.LAUF          ;  Ja:   Lauf
1857  A1                     AND     C               ;  Nein: Stop mit Bit
1858  4F                     LD      C.A             ;        Nr.1 = 0 in C

1859  C9                     RET                     ;Ende der Abfrage

185A  3E 02           LAUF:   LD      A.00000010B     ;Lauf mit Bit Nr.1 = 1
185C  B1                     OR      C               ;  im C-Register
185D  4F                     LD      C.A

185E  C9                     RET                     ;Ende der Abfrage

                      ;*****************************************************

                      RECHTS: ;Unterprogramm: 1-Bit im HL-Register zwischen
                              ;-------------  zwischen Bit Nr.1 und Bit Nr.6
                              ;               im Kreis um eine Stelle rechts
                              ;               herum schieben.
                              ;               Im L-Reg. (Port B) nach rechts
                              ;               Im H-Reg. (Port A) nach links

185F  F5                     PUSH    AF              ;Flag-Register retten

1860  AF                     XOR     A               ;Wenn H nicht leer.
1861  BC                     CP      H               ;  dann das 1-Bit
1862  20 15                  JR      NZ.HLI          ;  in H links schieben

1864  CB 3D                  SRL     L               ;1-Bit in L n. rechts
1866  3C                     INC     A               ;Steht in L 00000001
1867  BD                     CP      L               ;  (1 zu weit rechts)?
1868  28 05                  JR      Z.HBEG          ;  Ja:   (H) schieben
186A  7D                     LD      A.L             ;  Nein: (L) ausgeben
186B  D3 81                  OUT     (PIODB).A       ;        bei Port B.
186D  18 4E                  JR      FERTIG          ;Ende des Schiebens.

186F  AF              HBEG:  XOR     A               ;Ausgabe bei Port B
1870  D3 81                  OUT     (PIODB).A       ;  löschen.
1872  26 02                  LD      H.00000010B     ;1-Bit Nr.2 (rechts)
1874  7C                     LD      A.H             ;  über Port A
1875  D3 80                  OUT     (PIODA).A       ;  ausgeben.
1877  18 44                  JR      FERTIG          ;Ende des Schiebens.
1879  CB 24           HLI:   SLA     H               ;Steht in H 10000000
187B  3E 80                  LD      A.10000000B     ;  (1 zu weit links) ?
187D  BC                     CP      H
187E  28 05                  JR      Z.LBEG          ;  Ja:   (L) schieben
1880  7C                     LD      A.H             ;  Nein: (H) ausgeben
1881  D3 80                  OUT     (PIODA).A       ;        bei Port A.
1883  18 38                  JR      FERTIG          ;Ende des Schiebens.
```

```
1885  AF              LBEG:   XOR     A               ;Ausgabe bei Port A
1886  D3 80                   OUT     (PIODA).A       ;  löschen.
1888  2E 40                   LD      L.01000000B     ;1-Bit Nr.6 (links)
188A  7D                     LD      A.L             ;  über Port B
188B  D3 81                  OUT     (PIODB).A       ;  ausgeben.
188D  18 2E                  JR      FERTIG          ;Ende des Schiebens.

                        ;*************************************************

                        LINKS:  ;Unterprogramm: 1-Bit im HL-Register zwischen
                                ;-------------  zwischen Bit Nr.1 und Bit Nr.6
                                ;               im Kreis um eine Stelle LINKS
                                ;               herum schieben.
                                ;               Im L-Reg. (Port B) nach links
                                ;               Im H-Reg. (Port A) nach rechts

188F  F5                     PUSH    AF              ;Flag-Register retten

1890  AF                     XOR     A               ;Wenn H nicht leer.
1891  BC                     CP      H               ;  dann das 1-Bit
1892  20 16                  JR      NZ.HRE          ;  in H rechts schieben

1894  CB 25                  SLA     L               ;1-Bit in L n. links
1896  3E 80                  LD      A.10000000B     ;Steht in L 10000000
1898  BD                     CP      L               ;  (1 zu weit links) ?
1899  28 05                  JR      Z.HANF          ;  Ja:   (H) schieben
189B  7D                     LD      A.L             ;  Nein: (L) ausgeben
189C  D3 81                  OUT     (PIODB).A       ;         bei Port B.
189E  18 1D                  JR      FERTIG          ;Ende des Schiebens.

18A0  AF              HANF:  XOR     A               ;Ausgabe bei Port B
18A1  D3 81                  OUT     (PIODB).A       ;  löschen.
18A3  26 40                  LD      H.01000000B     ;1-Bit Nr.6 (links)
18A5  7C                     LD      A.H             ;  über Port A
18A6  D3 80                  OUT     (PIODA).A       ;  ausgeben.
18A8  18 13                  JR      FERTIG          ;Ende des Schiebens.

18AA  CB 3C           HRE:   SRL     H               ;Steht in H 00000001
18AC  3C                     INC     A               ;  (zu weit rechts) ?
18AD  BC                     CP      H
18AE  28 05                  JR      Z.LANF          ;  Ja:   (L) schieben
18B0  7C                     LD      A.H             ;  Nein: (H) ausgeben
18B1  D3 80                  OUT     (PIODA).A       ;        bei Port A.
18B3  18 08                  JR      FERTIG          ;Ende des Schiebens.
18B5  AF              LANF:  XOR     A               ;Ausgabe bei Port A
18B6  D3 80                  OUT     (PIODA).A       ;  löschen.
18B8  2E 02                  LD      L.00000010B     ;1-Bit Nr.2 (rechts)
18BA  7D                     LD      A.L             ;  über Port B
18BB  D3 81                  OUT     (PIODB).A       ;  ausgeben.

18BD  F1              FERTIG: POP     AF              ;Ende des Schiebens.
18BE  C9                     RET

                        ;*************************************************

                             END
```

```
                        ;*******************************************************
                        ;*                                                     *
                        ;*  Ausgabe eines Speicher-Inhalts im Micro-Professor  *
                        ;*          über eine Centronics-Schnittstelle         *
                        ;*                                                     *
                        ;*******************************************************

                                ;Definitionen

                        PIODA   EQU     80H             ;Daten,   Port A
                        PIODB   EQU     81H             ;Daten,   Port B
                        PIOBA   EQU     82H             ;Befehle, Port A
                        PIOBB   EQU     83H             ;Befehle, Port B

                        SPACE   EQU     20H             ;Leerzeichen
                        CR      EQU     0DH             ;Carriage Return
                        LF      EQU     0AH             ;Line Feed

                                ORG     1800H

                        ;*******************************************************

                        START:  ;PIO Ports A und B initialisieren

1800  CD 7F 18                   CALL    INIT

                                ;Speicherbereich festlegen

1803  21 00 18                   LD      HL,START        ;Anfangs-Adresse
1806  11 8F 18                   LD      DE,ENDADR       ;End-Adresse

                        ;*******************************************************

                        ADRESS: ;Hauptprogramm: Am Anfang der Druck-Zeile wird
                                ;-------------  die Adresse des ersten Bytes
                                ;               ausgegeben. Es folgen 16 Bytes

1809  7C                        LD      A,H             ;Adressen-HOB holen
180A  CD 47 18                  CALL    BYTDR           ;  und drucken.
180D  7D                        LD      A,L             ;Adressen-LOB holen
180E  CD 47 18                  CALL    BYTDR           ;  und drucken

1811  3E 20                     LD      A,SPACE         ;Zwei Leerzeichen aus
1813  CD 65 18                  CALL    DRUCK           ;  dem Akkumulator
1816  CD 65 18                  CALL    DRUCK           ;  drucken

1819  06 10                     INHALT: LD      B,16            ;Zähler für 16 Bytes

181B  7E                        LOOP:   LD      A,(HL)          ;Von (HL) adressiertes
181C  CD 47 18                          CALL    BYTDR           ;  Byte drucken

181F  3E 20                             LD      A,SPACE         ;Ein Leerzeichen
1821  CD 65 18                          CALL    DRUCK           ;  drucken
1824  CD 36 18                          CALL    OBEN            ;End-Adresse erreicht?
1827  23                                INC     HL              ;Nein: Nächste Adresse
1828  10 F1                             DJNZ    LOOP            ;16 Bytes ausdrucken
```

```
182A  3E 0D                   LD      A,CR          ;Nach 16 Bytes Carriage
182C  CD 65 18                CALL    DRUCK         ;  Return und Line Feed
182F  3E 0A                   LD      A,LF          ;  für nächste Zeile
1831  CD 65 18                CALL    DRUCK         ;  ausgeben
1834  18 D3                   JR      ADRESS        ;Nächste Zeile drucken

                        ;*****************************************************

                        OBEN:   ;Unterprogramm: Die aktuelle Adresse im HL-Regi-
                                ;               sterpaar wird mit der End-Adres-
                                ;               se im DE-Registerpaar vergli-
                                ;               chen. Wenn die End-Adresse er-
                                ;               reicht ist, wird CR/LF ausgege-
                                ;               ben und das Programm beendet,
                                ;               andernfalls fortgesetzt.

1836  7B                      LD      A,E           ;Wenn (E) ungleich (L),
1837  BD                      CP      L             ;  wird das Programm
1838  C0                      RET     NZ            ;  fortgesetzt.

1839  7A                      LD      A,D           ;Wenn (E)=(L), aber (H)
183A  BC                      CP      H             ;  ungleich (L). Pro-
183B  C0                      RET     NZ            ;  gramm fortsetzen
183C  3E 0D                   LD      A,CR          ;Sonst: Zeilenvorschub
183E  CD 65 18                CALL    DRUCK
1841  3E 0A                   LD      A,LF
1843  CD 65 18                CALL    DRUCK

1846  C7                      RST     0             ;Progamm Ende

                        ;*****************************************************
```

```
                        ;*****************************************************

                        BYTDR:  ;Unterprogramm: Ein Byte aus dem Akkumulator wird
                                ;-------------  als zwei aufeinanderfolgende
                                ;               ASCII-Bytes dem Drucker übergeben

1847  C5                        PUSH    BC              ;(BC) verwahren

1848  4F                        LD      C.A             ;Byte in C verwahren

1849  1F                        RRA                     ;Obere vier Bits
184A  1F                        RRA
184B  1F                        RRA
184C  1F                        RRA
184D  CD 5C 18                  CALL    HEXASC          ;  werden ASCII
1850  CD 65 18                  CALL    DRUCK           ;  und gedruckt

1853  79                        LD      A.C             ;Byte aus C zurückholen
1854  CD 5C 18                  CALL    HEXASC          ;  in ASCII wandeln
1857  CD 65 18                  CALL    DRUCK           ;  und drucken

185A  C1                        POP     BC              ;(BC) vom Stack holen

185B  C9                        RET                     ;Ende der Routine

                        HEXASC: ;Unterprogramm: Die unteren vier Bits im Akku-
                                ;               mulator werden ASCII

185C  E6 0F                     AND     00001111B       ;Maskieren
185E  C6 90                     ADD     A.90H           ;umwandeln
1860  27                        DAA
1861  CE 40                     ADC     A.40H
1863  27                        DAA

1864  C9                        RET                     ;Ende der Routine

                        ;*****************************************************
```

```
                        ;*****************************************************

                        DRUCK:  ;Unterprogramm: Diese Routine prüft den Drucker-
                                ;-------------  Status und wartet, bis der Druk-
                                ;               ker zur Übernahme eines ASCII-By-
                                ;               tes bereit ist.
                                ;               Wenn das der Fall ist, wird dem
                                ;               Drucker das ASCII-Byte aus dem
                                ;               Akkumulator übergeben.

1865  C5                                PUSH    BC              ;(BC) verwahren
1866  4F                                LD      C.A             ;(A) in C verwahren

1867  DB 81                      STAT:   IN      A.(PIODB)       ;Ist Ackn.     = 1
1869  E6 70                              AND       01110000B     ;    Busy      = 0
186B  EE 10                              XOR       00010000B     ;    Paper Out = 0 ?
186D  20 F8                              JR      NZ.STAT         ;Nein: Warten

186F  79                                 LD      A.C             ;Ja:   Byte holen und
1870  D3 80                              OUT     (PIODA).A       ;      an Drucker

1872  E3                                 EX      (SP).HL         ;Kurze Verzögerung
1873  E3                                 EX      (SP).HL

1874  3E 00                              LD      A,00000000B     ;STROBE auf 0,
1876  D3 81                              OUT     (PIODB).A
1878  3E 02                              LD      A,00000010B     ;       auf 1 schalten
187A  D3 81                              OUT     (PIODB).A

187C  79                                 LD      A.C             ;ASCII-Byte zurück holen
187D  C1                                 POP     BC              ;(BC) vom Stack holen
187E  C9                                 RET

                        ;*****************************************************
```

```
                        ;*****************************************************

                        INIT:   ;Unterprogramm: Programmierung der PIO-Ports A
                                ;-------------  und B.
                                ;               Über den Port A werden die ASCII-
                                ;               Bytes dem Drucker übergeben.
                                ;               Über die Bits Nr.1.4.5 und 6
                                ;               des Ports B wird das Handshake
                                ;               abgewickelt.

                                ;    Port A
                                ;           Mode Control Word

187F  3E CF                     LD      A.0CFH          ;Betriebsart 3
1881  D3 82                     OUT     (PIOBA).A

                                ;           I/O Register Control Word

1883  3E 00                     LD      A.00            ;Alle Anschlüsse Ausgänge
1885  D3 82                     OUT     (PIOBA).A

                                ;-----------------------------------------------

                                ;    Port B
                                ;           Mode Control Word

1887  3E CF                     LD      A,0CFH          ;Betriebsart 3
1889  D3 83                     OUT     (PIOBB).A

                                ;           I/O Register Control Word

188B  3E FD                     LD      A.11111101B     ;Bit Nr.1 = STROBE
                                                        ;Bit Nr.4 = Ackn.
                                                        ;Bit Nr.5 = Busy
                                                        ;Bit Nr.6 = Paper Out

188D  D3 83                     OUT     (PIOBB),A

188F  C9                ENDADR: RET

                        ;*****************************************************

                                END
```

Siehe S 19
Band: HPF-1 ab 12.8.84
Seite 1 170ff
INLALMP/EDI

```
                ;*******************************************************
                ;*                                                     *
                ;*      Über den PIO-Port B gesteuertes Lauflicht      *
                ;*   kann für eine Alarm-Routine unterbrochen werden   *
                ;*                                                     *
                ;*******************************************************

                        ;Definitionen

                PIODA   EQU     80H             ;Daten.   Port A
                PIODB   EQU     81H             ;Daten.   Port B
                PIOBA   EQU     82H             ;Befehle. Port A
                PIOBB   EQU     83H             ;Befehle. Port B

                        ORG     1800H

                ;*******************************************************

                START:  ;PIO Port A und B initialisieren

1800  CD 0D 18            CALL    INIT

                        ;Anfangs-Bitmuster für Ausgabe über Port B

1803  3E 01               LD      A.00000001B

                ;*******************************************************

                LOOP:   ;Hauptprogramm: Das Bitmuster im Akkumulator mit
                        ;------------  einem 1-Bit wird über den Port B
                        ;              ausgegeben. Das 1-Bit wird im
                        ;              Akkumulator nach links im Kreis
                        ;              geschoben. Nach Verzögerung er-
                        ;              folgt erneute Ausgabe über Port B

1805  D3 81               OUT     (PIODB).A       ;Ausgabe über Port B
1807  07                  RLCA                    ;Muster links schieben
1808  CD 31 18            CALL    WAIT            ;Verzögerung
180B  18 F8               JR      LOOP            ;Erneute Ausgabe

                ;*******************************************************
```

```
                        ;*****************************************************

                        INIT:   ;Unterprogramm: Programmierung der PIO-Ports A
                                ;-------------  und B und der CPU.
                                ;               Über den PIO-Port B werden 8 Bit
                                ;               als Bitmuster mit einem 1-Bit
                                ;               ausgegeben.
                                ;               Über die Bits Nr.0 und Nr.1 des
                                ;               PIO-Port A kann ein Interrupt an-
                                ;               gefordert werden.

                                ;    Port B
                                ;           Mode Control Word

180D  3E CF                     LD      A.0CFH          ;Betriebsart 3
180F  D3 83                     OUT     (PIOBB).A

                                ;           I/O Register Control Word

1811  3E 00                     LD      A.00H           ;Alle Anschlüsse Ausgänge
1813  D3 83                     OUT     (PIOBB).A

                                ;-----------------------------------------------

                                ;    Port A
                                ;           Mode Control Word

1815  3E CF                     LD      A.0CFH          ;Betriebsart 3
1817  D3 82                     OUT     (PIOBA).A

                                ;           I/O Register Control Word

1819  3E 03                     LD      A.00000011B     ;Bits Nr.0 und Nr.1
181B  D3 82                     OUT     (PIOBA).A       ;  werden Eingänge

                                ;           Interrupt Vector Word

181D  3E 58                     LD      A.LOW VALARM    ;LOB der Ablage-Adresse
181F  D3 82                     OUT     (PIOBA).A

                                ;           Interrupt Control Word

1821  3E B7                     LD      A.10110111B     ;Bit Nr.0-4: Kennung
                                                        ;Bit Nr.4: Maske folgt
                                                        ;Bit Nr.5: L-H - Interr.
                                                        ;Bit Nr.6: ODER
                                                        ;Bit Nr.7: Interrupt frei
1823  D3 82                     OUT     (PIOBA).A

                                ;           Mask Control Word

1825  3E FC                     LD      A.11111100B     ;Interr. über die Bits
1827  D3 82                     OUT     (PIOBA).A       ;  Nr.0 und Nr.1

                                ;-----------------------------------------------
```

```
                        ;-------------------------------------------------

                        ;    CPU-Programmierung

1829  3E 18             LD      A.HIGH VALARM   ;HOB der Ablage-Adresse
182B  ED 47             LD      I.A             ;  ins I-Register

182D  ED 5E             IM      2               ;Interrupt-Mode 2
182F  FB                EI                      ;IFF1 setzen

1830  C9                RET                     ;Ende der Routine

                      ;*****************************************************

                      WAIT:   ;Unterprogramm: Verzögerungsschleife
                              ;-------------  Das HL-Registerpaar wird vorge-
                              ;               laden und solange mit dem Inhalt
                              ;               1 des DE-Registerpaares inkremen-
                              ;               tiert. bis ein Überlauf im Car-
                              ;               ry-Flag entsteht.

1831  21 00 E8          LD      HL.0E800H       ;Verzögerungszeit
1834  11 01 00          LD      DE.1

1837  19                WAITL:  ADD     HL.DE           ;(HL) inkrementieren
1838  30 FD             JR      NC.WAITL        ;Überlauf? Ja: Weiter

183A  C9                RET                     ;Ende der Routine

                      ;*****************************************************

                              ;Definitionen

                      TONE1K  EQU     05DEH           ;1-kHz-Tongenerator
                      TONE2K  EQU     05E2H           ;2-kHz-Tongenerator

                      ALARM:  ;Interrupt-Service-Routine
                              ;------------------------
                              ;               Zwei Routinen aus dem Betriebs-
                              ;               Programm des Micro-Professors
                              ;               imitieren eine Alarm-Klingel.

183B  F5                PUSH    AF              ;Register-Inhalte auf
183C  E5                PUSH    HL              ;  dem Stack ablegen.
183D  D5                PUSH    DE

183E  06 18             LD      B.18H           ;Zähler für Perioden

1840  21 18 00          NEXT:   LD      HL.0018H        ;Perioden 1 kHz
1843  C5                PUSH    BC              ;(Zähler) verwahren
1844  CD DE 05          CALL    TONE1K          ;24 Perioden 1 kHz

1847  21 10 00          LD      HL.0010H        ;Perioden 2 kHz
184A  CD E2 05          CALL    TONE2K          ;16 Perioden 2 kHz
184D  C1                POP     BC              ;(Zähler) vom Stack
184E  10 F0             DJNZ    NEXT            ;Dieser Vorgang 24mal
```

```
1850  D1                    POP     DE          ;Register-Inhalte vom
1851  E1                    POP     HL          ;  Stack holen
1852  F1                    POP     AF

1853  FB                    EI                  ;IFF1 setzen
1854  ED 4D                 RETI                ;Ende der Routine

                      ;*****************************************************

                      ;Ablage-Speicherzellen für die Anfangsadresse
                      ;der Interrupt-Service-Routine ALARM

                      ORG     1858H           ;Adresse

1858  3B 18          VALARM: DW      ALARM       ;2 Byte mit ALARM-Start-
                                                ;  Adresse

                      ;*****************************************************

                      END
```

```
                        ;*******************************************************
                        ;*                                                     *
                        ;*      Über den PIO-Port B gesteuertes Lauflicht      *
                        ;*    kann durch Betätigung  abgefragter Tasten für    *
                        ;*       eine Alarm-Routine untebrochen werden         *
                        ;*                                                     *
                        ;*******************************************************

                                ;Definitionen

                        PIODA   EQU     80H             ;Daten.   Port A
                        PIODB   EQU     81H             ;Daten.   Port B
                        PIOBA   EQU     82H             ;Befehle. Port A
                        PIOBB   EQU     83H             ;Befehle. Port B

                                ORG     1800H

                        ;*******************************************************

                        START:  ;PIO Port A und B initialisieren

1800  CD 16 18                   call    init

                                ;Anfangs-Bitmuster für Ausgabe über Port B

1803  3E 01                     LD      A.00000001B

                        ;*******************************************************

                        LOOP:   ;Hauptprogramm: Das Bitmuster im Akkumulator mit
                                ;------------  einem 1-Bit wird über den Port B
                                ;              ausgegeben.
                                ;              Nach der Ausgabe werden die zum
                                ;              PIO-Port B gehörenden Tasten Nr.0
                                ;              und Nr.1 abgefragt. Wenn eine Ta-
                                ;              ste betätigt ist. wird die ALARM-
                                ;              Routine angesprungen.
                                ;              Das 1-Bit wird im Akkumulator
                                ;              nach links im Kreis geschoben.
                                ;              Nach Verzögerung erfolgt erneute
                                ;              Ausgabe über den Port B.
                                ;
1805  D3 81                     OUT     (PIODB).A       ;Ausgabe über Port B
1807  F5                        push    af              ;(Akku) verwahren
1808  DB 80                     in      a.(pioda)       ;Bitmuster von Port A
180A  E6 03                     and     00000011b       ;Nr.0.und Nr.1 maskieren
180C  C4 3B 18                  call    nz.alarm        ;Alarm. wenn Taste
180F  F1                        pop     af              ;(Akku) zurück holen
1810  07                        RLCA                    ;Muster links schieben
1811  CD 31 18                  CALL    WAIT            ;Verzögerung
1814  18 EF                     jr      loop            ;Erneute Ausgabe

                        ;*******************************************************
```

```
                        ;*****************************************************

                        INIT:   ;Unterprogramm: Programmierung der PIO-Ports A
                                ;------------  und B.
                                ;              Über den PIO-Port B werden 8 Bit
                                ;              als Bitmuster mit einem 1-Bit
                                ;              ausgegeben.
                                ;              Anschlüsse Nr.0 und Nr.1 des PIO-
                                ;              Ports A werden zum Erkennen von
                                ;              Tasten-Betätigungen als Eingänge
                                ;              programmiert.

                                ;    Port B
                                ;          Mode Control Word

1816  3E CF                            LD      A.0CFH          ;Betriebsart 3
1818  D3 83                            OUT     (PIOBB).A

                                ;          I/O Register Control Word

181A  3E 00                            LD      A.00H           ;Alle Anschlüsse Ausgänge
181C  D3 83                            OUT     (PIOBB).A

                                ;-----------------------------------------------

                                ;    Port A
                                ;          Mode Control Word

181E  3E CF                            LD      A.0CFH          ;Betriebsart 3
1820  D3 82                            OUT     (PIOBA).A

                                ;          I/O Register Control Word

1822  3E 03                            LD      A.00000011B     ;Bits Nr.0 und Nr.1
1824  D3 82                            OUT     (PIOBA).A       ;  werden Eingänge

1826  C9                               RET                     ;Ende der Routine

                        ;*****************************************************

                                ;Die folgenden Programmteile WAIT und ALARM be-
                                ;ginnen wie Lauflicht-Programm mit Interrupt.

                                org     1831h

                        WAIT:   ;Unterprogramm: Verzögerungsschleife
                                ;------------

1831  21 00 E8                        LD      HL.0E800H       ;Verzögerungszeit
1834  11 01 00                        LD      DE.1

1837  19                        WAITL: ADD     HL.DE           ;(HL) inkrementieren
1838  30 FD                            JR      NC.WAITL        ;Überlauf? Ja: Weiter

183A  C9                               RET                     ;Ende der Routine
                        ;*****************************************************
```

```
                        ;*****************************************************

                                ;Definitionen

                        TONE1K  EQU     05DEH           ;1-kHz-Tongenerator
                        TONE2K  EQU     05E2H           ;2-kHz-Tongenerator

                        ALARM:  ;Unterprogramm: Zwei Routinen aus dem Betriebs-
                                ;-------------  Programm des Micro-Professors
                                ;               imitieren eine Alarm-Klingel.

183B  F5                        PUSH    AF              ;Register-Inhalte auf
183C  E5                        PUSH    HL              ;  dem Stack ablegen.
183D  D5                        PUSH    DE

183E  06 18                     LD      B.18H           ;Zähler für Perioden

1840  21 18 00               NEXT:  LD      HL.0018H        ;Perioden 1 kHz
1843  C5                        PUSH    BC              ;(Zähler) verwahren
1844  CD DE 05                  CALL    TONE1K          ;24 Perioden 1 kHz

1847  21 10 00                  LD      HL.0010H        ;Perioden 2 kHz
184A  CD E2 05                  CALL    TONE2K          ;16 Perioden 2 kHz
184D  C1                        POP     BC              ;(Zähler) vom Stack

184E  10 F0                     DJNZ    NEXT            ;Dieser Vorgang 24mal

1850  D1                        POP     DE              ;Register-Inhalte vom
1851  E1                        POP     HL              ;  Stack holen
1852  F1                        POP     AF

1853  C9                        ret                     ;Ende der Routine

                        ;*****************************************************

                                END
```

Buchst.	Zeichen	Bitmuster	Byte	Neues Wort	Buchst.	Zeichen	Bitmuster	Byte	Neues Wort
A	.–	0110 0000	60	61	W	.––	0111 0000	70	71
B	–...	1000 1000	88	89	X	–..–	1001 1000	98	99
C	–.–.	1010 1000	A8	A9	Y	–.––	1011 1000	B8	B9
D	–..	1001 0000	90	91	Z	––..	1100 1000	C8	C9
E	.	0100 0000	40	41	0	–––––	1111 1100	FC	FD
F	..–.	0010 1000	28	29	1	.––––	0111 1100	7C	7D
G	––.	1101 0000	D0	D1	2	..–––	0011 1100	3C	3D
H	....	0000 1000	08	09	3	...––	0001 1100	1C	1D
I	..	0010 0000	20	21	4	....–	0000 1100	0C	0D
J	.–––	0111 1000	78	79	5	.....	0000 0100	04	05
K	–.–	1011 0000	B0	B1	6	–....	1000 0100	84	85
L	.–..	0100 1000	48	49	7	––...	1100 0100	C4	C5
M	––	1110 0000	E0	E1	8	–––..	1110 0100	E4	E5
N	–.	1010 0000	A0	A1	9	––––.	1111 0100	F4	F5
O	–––	1111 0000	F0	F1	.	.–.–.–	0101 0110	56	57
P	.––.	0110 1000	68	69	,	––..––	1100 1110	CE	CF
Q	––.–	1101 1000	D8	D9	:	–––...	1110 0010	E2	E3
R	.–.	0101 0000	50	51	?	..––..	0011 0010	32	33
S	...	0001 0000	10	11	–	–....–	1000 0110	86	87
T	–	1100 0000	C0	C1	/	–..–.	1001 0100	94	95
U	..–	0011 0000	30	31	+	.–.–.	0101 0100	54	55
V	...–	0001 1000	18	19					

Bild L 30.1
Diese Tabelle der wichtigsten Morsezeichen zeigt die Bitmuster und Bytes, aus denen das Morse-Programm die Zeichen generiert.

```
                ;*******************************************************
                ;*                                                     *
                ;*                  Morse-Programm                     *
                ;*                                                     *
                ;* Ein fest vorgegebener Text  wird repitierend in Form *
                ;*           von Morsezeichen ausgegeben.              *
                ;*                                                     *
                ;*  Die  Ausgabe  kann  für  zwei  unterschiedliche In- *
                ;*    terrupt-Service-Routinen unterbrochen werden.    *
                ;*                                                     *
                ;*******************************************************

                        ;Definitionen:

                PIODA   EQU     80H             ;Daten.   Port A
                PIODB   EQU     81H             ;Daten.   Port B
                PIOBA   EQU     82H             ;Befehle. Port A
                PIOBB   EQU     83H             ;Befehle. Port B

                DIGIT   EQU     02H             ;8255. Port A

                        ;Routinen-Adressen im Betriebsprogramm des
                        ;Micro-Professors:

                TONE1K  EQU     05DEH           ;1-kHz-Tongenerator
                TONE2K  EQU     05E2H           ;2-kHz-Tongenerator

                        ORG     1800H

                ;*******************************************************

                START:  ;Initialisierung der PIO-Ports A und B
                        ;und der CPU

1800  CD 55 18            CALL    INIT

                MORSE:  ;Hauptprogramm: Die im Speicher ab dem Label
                        ;-------------  TEXT bis zum Byte 00 abgelegten.
                        ;               codierten Morsezeichen werden
                        ;               über den Lautsprecher des Micro-
                        ;               Professors ausgegeben. Die Aus-
                        ;               gabe wird zyklisch wiederholt.

1803  21 F0 18            LD      HL.TEXT         ;(HL) zeigt auf TEXT

1806  7E          BYTE:   LD      A.(HL)          ;Morse-Byte holen
1807  FE 00               CP      00              ;Ist es 00?
1809  28 F8               JR      Z.MORSE         ;  Ja: Neue TEXT-Ausgabe
180B  CD 85 18            CALL    KEY             ;  Nein: Zeichen ausgeben
180E  18 F6               JR      BYTE            ;Nächstes Zeichen

                ;*******************************************************
```

```
                        ;*****************************************************

                        ALARM:  ;Interrupt-Service-Routine
                                ;------------------------
                                ;               Zwei Routinen aus dem Betriebs-
                                ;               Programm des Micro-Professors
                                ;               imitieren eine Alarm-Klingel.
                                ;
                                ;               Die Routine hinterläßt alle Re-
                                ;               gister-Inhalte unverändert.

1810  00                                NOP                     ;Reserviert

1811  F5                                PUSH    AF              ;Register-Inhalte auf
1812  C5                                PUSH    BC              ;  dem Stack ablegen.
1813  E5                                PUSH    HL
1814  D5                                PUSH    DE

1815  06 18                             LD      B,18H           ;Zähler für Perioden

1817  21 18 00                  NEU:    LD      HL,0018H        ;Perioden 1 kHz
181A  C5                                PUSH    BC              ;(Zähler) verwahren
181B  CD DE 05                          CALL    TONE1K          ;24 Perioden 1 kHz
181E  00                                NOP
181F  00                                NOP
1820  21 10 00                          LD      HL,0010H        ;16 Perioden 2 kHz
1823  CD E2 05                          CALL    TONE2K
1826  C1                                POP     BC              ;(Zähler) vom Stack
1827  10 EE                             DJNZ    NEU             ;Dieser Vorgang 24mall

1829  D1                                POP     DE              ;Register-Inhalte vom
182A  E1                                POP     HL              ;  Stack holen
182B  C1                                POP     BC
182C  F1                                POP     AF

182D  FB                                EI                      ;IFF1 setzen
182E  ED 4D                             RETI                    ;Ende der Routine

                        ;*****************************************************
```

```
                        ;*****************************************************

                        SOS:    ;Interrupt-Service-Routine
                                ;------------------------
                                ;               Die im Speicher ab TEXT1 bis zum
                                ;               Byte 00 abgelegten. codierten
                                ;               Morsezeichen werden über den
                                ;               Lautsprecher ausgegeben. Nach
                                ;               einmaliger Ausgabe des Textes ist
                                ;               die Routine beendet.
                                ;
                                ;               Die Routine hinterläßt alle Re-
                                ;               gister-Inhalte und ein in der
                                ;               Ausgabe-Routine verwendetes Flag
                                ;               unverändert.

1830  00                                NOP                     ;Reserviert

1831  F5                                PUSH    AF              ;Register-Inhalte auf
1832  C5                                PUSH    BC              ;  dem Stack ablegen
1833  D5                                PUSH    DE
1834  E5                                PUSH    HL

1835  3A E6 18                          LD      A.(FLAG)        ;(Flag) verwahren
1838  32 E7 18                          LD      (FLAG+1).A

183B  21 2D 19                          LD      HL.TEXT1        ;(HL) zeigt auf TEXT1

183E  7E                        BYTE1:  LD      A.(HL)          ;Morse-Byte holen
183F  FE 00                             CP      00              ;Ist es 00?
1841  28 05                             JR      Z.ENDINT        ;  Ja: Routinen-Ende
1843  CD 85 18                          CALL    KEY             ;  Nein: Zeichen ausgeben
1846  18 F6                             JR      BYTE1           ;Nächstes Zeichen

                                        ;---------------------------------

1848  3A E7 18                 ENDINT: LD      A.(FLAG+1)      ;Ursprünglichen Inhalt
184B  32 E6 18                          LD      (FLAG).A        ;  des Flags
184E  E1                                POP     HL              ;und alle Register-In-
184F  D1                                POP     DE              ;  halte wieder her-
1850  C1                                POP     BC              ;  stellen.
1851  F1                                POP     AF

1852  FB                                EI                      ;IFF1 setzen
1853  ED 4D                             RETI                    ;Ende der Routine
                        ;*****************************************************
```

```
                        ;*****************************************************

                        INIT:   ;Unterprogramm: Programmierung der PIO-Ports A
                                ;-------------  A und B und der CPU.
                                ;               Über die Tasten Nr.0 und Nr.1
                                ;               des Ports A wird der Alarm-Inter-
                                ;               rupt aufgerufen.
                                ;               Über die Tasten Nr.6 und Nr.7
                                ;               des Ports B wird der SOS-Inter-
                                ;               rupt aufgerufen.
                                ;
                                ;    Port A
                                ;           Mode Control Word

1855  3E CF                             LD      A.0CFH          ;Betriebsart 3
1857  D3 82                             OUT     (PIOBA).A

                                ;           I/O Register Control Word

1859  3E 03                             LD      A.00000011B     ;Bits Nr.0 und Nr.1
185B  D3 82                             OUT     (PIOBA).A       ;  werden Eingänge

                                ;           Interrupt Vector Word

185D  3E E8                             LD      A.LOW VALARM    ;LOB der Ablage-Adresse
185F  D3 82                             OUT     (PIOBA).A       ;  für die ALARM-Routine

                                ;           Interrupt Control Word

1861  3E B7                             LD      A.10110111B     ;Bits Nr.0-3: Kennung
                                                                ;Bit Nr.4: Maske folgt
                                                                ;Bit Nr.5: L-H - Interr.
                                                                ;Bit Nr.6: ODER
                                                                ;Bit Nr.7: Interrupt frei
1863  D3 82                             OUT     (PIOBA).A

                                ;           Mask Control Word

1865  3E FC                             LD      A.11111100B     ;Interr. über die Bits
1867  D3 82                             OUT     (PIOBA).A       ;  Nr.0 und Nr.1

                                ;-----------------------------------------------

                                ;    Port B
                                ;           Mode control Word

1869  3E CF                             LD      A.0CFH          ;Betriebsart 3
186B  D3 83                             OUT     (PIOBB).A

                                ;           I/O Register Control Word

186D  3E C0                             LD      A.11000000B     ;Bits Nr.6 und Nr.7
186F  D3 83                             OUT     (PIOBB).A       ;  werden Eingänge
```

```
                        ;          Interrupt Vector Word

1871  3E EA             LD       A.LOW VSOS      ;LOB der Ablage-Adresse
1873  D3 83             OUT      (PIOBB).A       ;  für die SOS-Routine

                        ;          Interrupt Control Word

1875  3E B7             LD       A.10110111B     ;Bits Nr.0-3: Kennung
                                                  ;Bit Nr.4: Maske folgt
                                                  ;Bit Nr.5: L-H - Interr.
                                                  ;Bit Nr.6: ODER
                                                  ;Bit Nr.7: Interrupt frei
1877  D3 83             OUT      (PIOBB).A

                        ;          Mask Control Word

1879  3E 3F             LD       A.00111111B     ;Interr. über die Bits
187B  D3 83             OUT      (PIOBB).A       ;  Nr.6 und Nr.7

                        ;-----------------------------------------------

                        ;    CPU-Programmierung

187D  3E 18             LD       A.18H           ;HOB der Ablage-Adressen
187F  ED 47             LD       I.A             ;  ins I-Register

1881  ED 5E             IM       2               ;Interrupt-Mode 2
1883  FB                EI                       ;IFF1 setzen

1884  C9                RET                      ;Ende der Routine

                      ;*****************************************************
```

```
                        ;*****************************************************

                        KEY:    ;Unterprogramm: Das Morse-Byte steht im Akku.
                                ;-------------  Bit Nr.0 = 1 im Morse-Byte kenn-
                                ;               zeichnet ersten Buchstaben eines
                                ;               Worts mit längerer Pause vor der
                                ;               Ausgabe des Zeichens.
                                ;               Das Morse-Byte im Akkumulator
                                ;               wird Bit-weise nach links ins
                                ;               Carry-Flag geschoben. Je nach
                                ;               Wert wird ein kurzes oder langes
                                ;               akustisches Zeichen-Element aus-
                                ;               gegeben. Byte 80H im Akku bedeu-
                                ;               tet Terminator = Bit Nr.7. also
                                ;               Zeichen-Ende. Es folgt eine Pause

1885  E6 01                            AND     00000001B       ;Bit Nr.0 maskieren
1887  28 05                            JR      Z.BIT           ;Wenn Bit Nr.0 = 1.
1889  06 04                     WORT:  LD      B.04H           ;  dann längere
188B  CD BB 18                        CALL    PAUSE           ;  Wort-Pause
188E  7E                         BIT:  LD      A.(HL)          ;Morse-Byte nochm. holen
188F  E6 FE                            AND     11111110B       ;Bit Nr.0 ausblenden
1891  CB 27                     SCHIEB: SLA     A               ;Bit Nr.7 wird Carry-Bit
1893  30 0D                            JR      NC.KURZ         ;CY=0: Kurzes Element
1895  38 0F                            JR      C.LANG          ;CY=1: Langes Element
1897  FE 80                      TERM: CP      80H             ;Terminator = Bit Nr.7?
1899  20 F6                            JR      NZ.SCHIEB       ;  Nein: Nächstes Bit
189B  06 02                      ZEICH: LD      B.02H           ;  Ja:   Zeichen Ende
189D  CD BB 18                        CALL    PAUSE           ;        und Pause
18A0  23                              INC     HL              ;Zeiger auf nächst. Byte

18A1  C9                              RET                     ;Ende der Routine

                                ;-----------------------------------------------

                        KURZ:   ;Ausgabe der Zeichen-Elemente und der Pause bis
                                ;zum folgenden Zeichenelement.

18A2  06 01                           LD      B.01H           ;Dauer d. kurzen Z.-Elem.
18A4  18 02                           JR      SIGN
18A6  06 03                      LANG: LD      B.03            ;Dauer d. langen Z.-Elem.
18A8  E5                         SIGN: PUSH    HL              ;(HL) verwahren
18A9  21 E6 18                        LD      HL.FLAG         ;Zeiger auf Flag
18AC  36 FF                           LD      (HL).0FFH       ;  Flag für Ton
18AE  CD C6 18                        CALL    ZEIT            ;Ausgabe
18B1  06 01                           LD      B.01H           ;Abstand
18B3  36 7F                           LD      (HL).7FH        ;  Flag für Abstand
18B5  CD C6 18                        CALL    ZEIT            ;Ausgabe
18B8  E1                              POP     HL              ;(HL) vom Stack holen
18B9  18 DC                           JR      TERM            ;Zeichen-Ende?

                                ;-----------------------------------------------
```

```
                                ;-----------------------------------------------

                        PAUSE:  ;Unterprogramm: Ausgabe einer längeren Pause zwi-
                                ;               schen zwei Worten. wenn im Morse-
                                ;               Byte das Bit Nr.0 den Wert 1 hat.

18BB  E5                        PUSH    HL              ;(HL) verwahren
18BC  21 E6 18                  LD      HL.FLAG         ;Zeiger auf Flag
18BF  36 7F                     LD      (HL).7FH        ;Flag für Abstand
18C1  CD C6 18                  CALL    ZEIT            ;Ausgabe
18C4  E1                        POP     HL              ;(HL) vom Stack holen

18C5  C9                        RET                     ;Ende der Routine

                                ;-----------------------------------------------

                        ZEIT:   ;Unterprogramm: Ausgabe eines Zeichen-Elements
                                ;               bzw. eines Abstands entsprechen-
                                ;               der Länge.

18C6  F5                        PUSH    AF              ;(AF) verwahren
18C7  2B                UNIT:   DEC     HL              ;Zeiger auf SPEED
18C8  4E                        LD      C.(HL)          ;(SPEED) ins C-Register
18C9  23                        INC     HL              ;Zeiger auf Flag

18CA  7E                PERIOD: LD      A.(HL)          ;Positive Halbwelle
18CB  D3 02                     OUT     (DIGIT).A       ;  an Lautsprecher
18CD  CD DE 18                  CALL    WAIT            ;Dauer

18D0  3E 7F                     LD      A.7FH           ;Negative Halbwelle
18D2  D3 02                     OUT     (DIGIT).A       ;  an Lautsprecher
18D4  CD DE 18                  CALL    WAIT            ;Dauer

18D7  0D                        DEC     C               ;Länge d. Z.-Elements
18D8  20 F0                     JR      NZ.PERIOD       ;Noch 'ne Periode

18DA  10 EB                     DJNZ    UNIT            ;Mehrf. bei langem
                                                        ;  Zeichen-Element

18DC  F1                        POP     AF              ;(AF) vom Stack holen

18DD  C9                        RET                     ;Ende der Routine

                                ;-----------------------------------------------

                        WAIT:   ;Unterprogramm: Zeitelemente der Halbwellen-Dauer

18DE  C5                        PUSH    BC              ;(BC) verwahren
18DF  06 80                     LD      B.80H           ;Dauer
18E1  10 FE             LOOP:   DJNZ    LOOP
18E3  C1                        POP     BC              ;(BC) vom Stack holen

18E4  C9                        RET                     ;Ende der Routine

                                ;-----------------------------------------------
```

```
                                ;-----------------------------------------------

                                ;Zwei Speicherzellen für die Ablage von im Pro-
                                ;gramm verwendeten Werten

18E5  20                SPEED:  DB      20H             ;Morse-Geschwindigkeit
                        FLAG:   DS      00              ;Reserviert für die Ken-
                                                        ;nung v. Ton oder Pause

                        ;*****************************************************

                                ;Vektorfeld:    Ablage-Speicherzellen für die An-
                                ;----------     fangs-Adressen der Interrupt-Ser-
                                ;               vice-Routinen

                                ORG     18E8H

18E8  10 18             VALARM: DW      ALARM
18EA  30 18             VSOS:   DW      SOS

                        ;*****************************************************

                                ORG     18F0H

                        TEXT:   ;Ablage der codierten Morse-Bytes für den norma-
                                ;len Text

18F0  C1 40 A8 08 A0 20 10 A8           DB      0C1H.40H.0A8H.08H.0A0H.20H.10H.0A8H
18F8  08 40 10 49 40 08 50 20           DB      08H.40H.10H.49H.40H.08H.50H.20H
1900  A0 10 C0 20 C0 30 C0 91           DB      0A0H.10H.0C0H.20H.0C0H.30H.0C0H.91H
1908  50 56 A9 08 50 20 10 C0           DB      50H.56H.0A9H.08H.50H.20H.10H.0C0H
1910  20 60 A0 20 B1 F0 A0 10           DB      20H.60H.0A0H.20H.0B1H.0F0H.0A0H.10H
1918  C0 60 A0 C8 95 89 F0 90           DB      0C0H.60H.0A0H.0C8H.95H.89H.0F0H.90H
1920  40 A0 10 40 40 55 90 78           DB      40H.0A0H.10H.40H.40H.55H.90H.78H
1928  7C 18 A8 55                       DB      7CH.18H.0A8H.55H
192C  00                                DB      00              ;Ende-Markierung

                        TEXT1:  ;Ablage der codierten Morse-Bytes für dreimal
                                ;SOS in der SOS-Interrupt-Service-Routine

192D  11 F0 10 11 F0 10 11 F0           DB      11H.0F0H.10H.11H.0F0H.10H.11H.0F0H
1935  10                                DB      10H
1936  00                                DB      00              ;Ende-Markierung

                        ;*****************************************************

                                END
```

HEDUINMP/EDI D 404 (P K 1)

```
                ;*******************************************************
                ;*                                                     *
                ;*    Druck-Routine und PIO-Initialisierung für eine   *
                ;*              Centronics-Schnittstelle.              *
                ;*                                                     *
                ;*     Ein ASCII-Byte aus dem Akkumulator wird dem     *
                ;*  Drucker übergeben, wenn dieser seine Bereitschaft  *
                ;*    mit einer Interrupt-Anforderung gemeldet hat.    *
                ;*                                                     *
                ;*     Dieses Programm ab der Adresse 1865H ersetzt    *
                ;*   den enstsprechenden Teil des  HEX-DUMP-Programms  *
                ;*      ab der Adresse 1865H  auf der Seite  L 20.     *
                ;*                                                     *
                ;*******************************************************


                        ;Definitionen

                PIODA   EQU     80H     ;Daten.   Port A
                PIODB   EQU     81H     ;Daten.   Port B
                PIOBA   EQU     82H     ;Befehle. Port B
                PIOBB   EQU     83H     ;Befehle. Port B

                        ORG     1865H

                ;*******************************************************

                DRUCK:  ;Unterprogramm: Das ASCII-Byte aus dem Akkumula-
                        ;------------  tor wird dem Drucker übergeben.
                        ;              Es folgt ein STROBE-Signal.
                        ;              Danach wartet die CPU auf einen
                        ;              Interrupt vom Drucker. Erst dann
                        ;              wird das nächstfolgende Byte
                        ;              übergeben.

1865  D3 80                     OUT     (PIODA).A       ;Byte an Drucker

1867  3E 00                     LD      A.00000000B     ;STROBE
1869  D3 81                     OUT     (PIODB).A
186B  3E 02                     LD      A.00000010B
186D  D3 81                     OUT     (PIODB).A

186F  76                        HALT                    ;CPU wartet
1870  C9                        RET                     ;Interr. ist gekommen

                ;*******************************************************
```

```
                        ;*****************************************************

                INIT:   ;Unterprogramm: Programmierung des PIO-Ports A
                        ;------------   zur Ausgabe der ASCII-Bytes.
                        ;               Interrupt-Programmierung des PIO-
                        ;               Ports B für Drucker-STROBE und
                        ;               und Interrupt.
                        ;               Interrupt-Programmierung der CPU

                        ;    Port A
                        ;           Mode Control Word

1871  3E CF                    LD      A.0CFH          ;Betriebsart 3
1873  D3 82                    OUT     (PIOBA).A

                        ;           I/O Register Control Word

1875  3E 00                    LD      A.00000000B     ;Alle Anschlüsse Ausgänge
1877  D3 82                    OUT     (PIOBA).A

                        ;-----------------------------------------------

                        ;    Port B
                        ;           Mode Control Word

1879  3E CF                    LD      A.0CFH          ;Betriebsart 3
187B  D3 83                    OUT     (PIOBB).A

                        ;           I/O Register Control Word

187D  3E 60                    LD      A.01100000B     ;Bit Nr.1 = STROBE
                                                       ;Bit Nr.5 = Busy
                                                       ;Bit Nr.6 = Paper Out
187F  D3 83                    OUT     (PIOBB).A

                        ;           Interrupt Vector Word

1881  3E 98                    LD      A.LOW VWAIT     ;Interrupt-Vector LOB
1883  D3 83                    OUT     (PIOBB).A

                        ;           Interrupt Control Word

1885  3E D7                    LD      A.11010111B     ;Bits Nr. 0-3: Kennung
                                                       ;Bit Nr.4: Maske folgt
                                                       ;Bit Nr.5: H-L - Interr.
                                                       ;Bit Nr.6: UND
                                                       ;Bit Nr.7: Interrupt frei
1887  D3 83                    OUT     (PIOBB).A

                        ;           Mask Control Word

1889  3E 9F                    LD      A.10011111B     ;Interr. über die Bits
188B  D3 83                    OUT     (PIOBB).A       ;  Nr.5 und Nr.6

                        ;-----------------------------------------------
```

```
                        ;-----------------------------------------------

                        ;    CPU-Programmierung

188D  3E 18             LD      A.HIGH VWAIT    ;Interrupt-Vektor HOB
188F  ED 47             LD      I.A             ;  ins I-Register

1891  ED 5E             IM      2               ;Interrupt-Mode 2
1893  FB                EI                      ;IFF1 setzen

1894  C9                RET                     ;Ende der Routine

                    ;*****************************************************

                    WAIT:   ;Interrupt Service Routine
                            ;------------------------
                            ;               Nach Interrupt erfolgt unmittel-
                            ;               bar ein Rücksprung nach RET in
                            ;               der Druck-Routine und danach die
                            ;               Bereitstellung des nächstfolgen-
                            ;               den ASCII-Bytes.

1895  FB                EI                      ;IFF1 setzen
1896  ED 4D             RETI                    ;Ende der Routine

                    ;*****************************************************

                            ;Vektorfeld:    Ablage-Speicherzellen fÜr die An-
                            ;----------     fangsadressen der Interrupt-Ser-
                            ;               vice-Routine

                            ORG 1898H

1898  95 18            VWAIT:  DW      WAIT

                    ;*****************************************************

                    ENDADR: END
```

```
                        ;*******************************************************
                        ;*    Druck-Routine und PIO-Initialisierung für eine   *
                        ;*       Centronics-Schnittstelle.  (PIO Mode 0.)      *
                        ;*                                                     *
                        ;*     Ein ASCII-Byte aus dem  Akkumulator wird dem    *
                        ;*  Drucker übergeben.  wenn dieser seine Bereitschaft *
                        ;*     mit einer Interrupt-Anforderung gemeldet hat.   *
                        ;*                                                     *
                        ;*      Dieses Programm ab der Adresse 1865 ersetzt    *
                        ;*    den entsprechenden Teil des  HEX-DUMP-Programms  *
                        ;*        ab der Adrese 1865H auf der Seite L XX.      *
                        ;*******************************************************


                               ;Definitionen

                        PIODA  EQU     80H     ;Daten,  Port A
                        PIOBA  EQU     82H     ;Befehle, Port A

                               ORG     1865H

                        ;*******************************************************

                        DRUCK: ;Unterprogramm: Das ASCII-Byte aus (A) an den
                               ;------------- Drucker. Übergabe des nächstfol-
                                              genden Bytes erst nach Interrupt.

1865  D3 80                   OUT     (PIODA),A       ;Byte an Drucker
1867  76                     HALT                    ;CPU wartet
1868  C9                     RET                     ;Interr. ist gekommen

                        ;*******************************************************

                        INIT:  ;Unterprogramm: Programmierung des PIO-Ports A
                               ;------------- zur Ausgabe der ASCII-Bytes und
                               ;              für Handshake im Mode 0.
                               ;              Interrupt-Programmierung der CPU

                               ;    Port A
                               ;          Mode Control Word

1869  3E 0F                   LD      A,0FH           ;Betriebsart 0
186B  D3 82                   OUT     (PIOBA),A

                               ;          Interrupt Vector Word

186D  3E 80                   LD      A,LOW VWAIT     ;Interrupt-Vektor LOB
186F  D3 82                   OUT     (PIOBA),A

                               ;          Interrupt Disable Word

1871  3E 83                   LD      A,83H           ;
1873  D3 82                   OUT     (PIOBA),A

                               ;---------------------------------------------
```

```
                        ;-----------------------------------------------

                        ;    CPU-Programmierung

1875  3E 18             LD      A.HIGH VWAIT    ;Interrupt-Vektor HOB
1877  ED 47             LD      I.A             ;  ins I-Register

1879  ED 5E             IM      2               ;Interrupt-Mode 2
187B  FB                EI                      ;IFF1 setzen

187C  C9                RET                     ;Ende der Routine

                    ;*****************************************************

                    WAIT:   ;Interrupt-Service-Routine
                            ;------------------------
                            ;               Nach Interrupt erfolgt unmittel-
                            ;               bar ein Rücksprung nach RET in
                            ;               der Druck-Routine und danach die
                            ;               Bereitstellung des nächstfolgen-
                            ;               den ASCII-Bytes.

187D  FB                    EI              ;IFF1 setzen
187E  ED 4D                 RETI            ;Ende der Routine

                    ;*****************************************************

                            ;Vektorfeld:    Ablage-Speicherzellen für die An-
                            ;----------     fangsadressen der Interrupt-Ser-
                            ;               vice-Routine

                            ORG 1880H

1880  7D 18           VWAIT: DW      WAIT

                    ;*****************************************************

                    ENDADR: END
```

```
                        ;*******************************************************
                        ;*                  Ereignis-Zähler                    *
                        ;*                  ---------------                    *
                        ;*    Der CTC zählt eine voreingestellte Anzahl von    *
                        ;*    Tasten-Betätigungen. Wenn die Anzahl erreicht    *
                        ;*    ist. wird ein  akustisches Signal ausgegeben.    *
                        ;*******************************************************

                               ;Definitionen

                        CTC0   EQU     40H     ;CTC Kanal 0
                        TONE1K EQU     05DEH   ;1-kHz-Periode
                        TONE2K EQU     05E2H   ;2-kHz-Periode

                               ORG     1800H

                        ;*******************************************************
1800  CD 06 18         ANFANG: CALL    INIT            ;Programmierung von CTC
                                                       ;und CPU für Interrupt
                        ZAEHL: ;Hauptprogramm: Die CPU wartet im HALT-Zustand
                               ;------------  auf Zähl-Interrupts vom CTC.

1803  76                       HALT                    ;Auf Interrupt warten
1804  18 FD                    JR      ZAEHL           ;Nächster Interrupt

                        ;*******************************************************
                        INIT:  ;Unterprogramm: Der CTC wird als Zähler für fünf
                               ;------------  Ereignisse programmiert.

                               ;    CTC Kanal 0
                               ;              Channel Control Word

1806  3E D7                   LD      A.11010111B     ;Bit Nr.0: Kennung
                                                       ;Bit Nr.1: Reset
                                                       ;Bit Nr.2: Folgt Zeitk.
                                                       ;Bit Nr.3: (Don't Care)
                                                       ;Bit Nr.4: 0-1 - Flanke
                                                       ;Bit Nr.5: (Don't Care)
                                                       ;Bit Nr.6: Counter Mode
                                                       ;Bit Nr.7: Interrupt frei
1808  D3 40                    OUT     (CTC0).A

                               ;              Time Constant Word

180A  3E 05                    LD      A.5             ;5 Ereignisse
180C  D3 40                    OUT     (CTC0).A

                               ;              Interrupt Vector Word

180E  3E 30                    LD      A.LOW VSIGNAL   ;Interrupt Vektor LOB
1810  D3 40                    OUT     (CTC0).A

                               ;-----------------------------------------
```

```
                        ;-----------------------------------------------
                        CPUINI
                        ;    CPU-Programmierung

1812  3E 18             LD      A.HIGH VSIGNAL  ;Interrupt-Vektor HOB
1814  ED 47             LD      I.A             ;  ins I-Register

1816  ED 5E             IM      2               ;Interrupt-Mode 2
1818  FB                EI                      ;IFF1-Flag setzen

1819  C9                RET                     ;Ende der Routine

                    ;*****************************************************

                    SIGNAL: ;Unterprogramm: Mit Routinen aus dem Betriebs-
                            ;------------  Programm wird eine Klingel pro-
                            ;              grammiert.

181A  06 18                 LD      B.24            ;24 Klingel-Anschläge

181C  21 18 00              NEXT:   LD      HL.24           ;24 Perioden 1 kHz
181F  C5                    PUSH    BC              ;(B) verwahren
1820  CD DE 05              CALL    TONE1K
1823  21 10 00              LD      HL.16           ;16 Perioden 2 kHz
1826  CD E2 05              CALL    TONE2K
1829  C1                    POP     BC              ;(B) vom Stack holen
182A  10 F0                 DJNZ    NEXT            ;Schleife 24 mal

182C  FB                    EI                      ;IFF1 setzen
182D  ED 4D                 RETI                    ;Ende der Routine

                    ;*****************************************************

                            ORG     1830H

                            ;Vektorfeld     Ablage-Speicherzellen für die An-
                            ;               fangsadresse der Interrupt-Ser-
                            ;               vice-Routine

1830  1A 18               VSIGNAL:DW      SIGNAL          ;CTC-Interrupt

                    ;*****************************************************

                            END
```

```
                        ;*******************************************************
                        ;*                                                     *
                        ;*                Interrupt-Lauflicht                  *
                        ;*               ----------------------                *
                        ;*   Ein Lauflicht wird über Kanal B der PIO mit In-   *
                        ;*         terrupt-Impulsen vom CTC gesteuert.         *
                        ;*                                                     *
                        ;*******************************************************

                                ;Definitionen

0040                    CTC0    EQU     40H     ;CTC Kanal 0

0081                    PIODB   EQU     81H     ;Befehle PIO-Port A
0083                    PIOBB   EQU     83H     ;Befehle PIO-Port B

                                ORG     1800H

                        ;*******************************************************

1800    CD 06 18        ANFANG: CALL    INIT            ;Programmierung von PIO.
                                                        ;CTC und CPU für Interr.

1803                    BLINK:  ;Hauptprogramm: Die CPU wartet im HALT-Zustand
                                ;-------------  auf Lauflicht-Interrupts vom CTC.

1803    76                      HALT                    ;Auf Interrupt warten
1804    18 FD                   JR      BLINK           ;Nächster Interrupt

                        ;*******************************************************

1806                    INIT:   ;Unterprogramm: Programmierung des PIO-Ports B
                                ;-------------  Programmierung des CTC-Kanals 0
                                ;               Interrupt-Programmierung der CPU

                                ;    PIO-Port B
                                ;               Mode Control Word

1806    3E CF                   LD      A.0CFH          ;Betriebsart 3
1808    D3 83                   OUT     (PIOBB).A

                                ;               I/O Register Control Word

180A    3E 00                   LD      A.00000000B     ;Alle Anschlüsse Ausgänge
180C    D3 83                   OUT     (PIOBB).A

                                ;-----------------------------------------------
```

```
                        ;-----------------------------------------------

                        ;   CTC Kanal 0
                        ;              Channel Control Word

180E   3E B7            LD      A.10110111B     ;Bit Nr.0: Kennung
                                                ;Bit Nr.1: RESET
                                                ;Bit Nr.2: Folgt Zeitk.
                                                ;Bit Nr.3: Autom. Start
                                                ;Bit Nr.4: (Don't Care)
                                                ;Bit Nr.5: Teiler = 256
                                                ;Bit Nr.6: Timer Mode
                                                ;Bit Nr.7: Interrupt frei
1810   D3 40            OUT     (CTC0).A

                        ;              Zeitkonstante

1812   3E FF            LD      A.0FFH
1814   D3 40            OUT     (CTC0).A

                        ;              Interrupt Vector Word

1816   3E 38            LD      A.LOW VINTCT0   ;Interrupt-Vektor LOB
1818   D3 40            OUT     (CTC0).A

                        ;-----------------------------------------------

                        ;   CPU-Programmierung

181A   3E 18            LD      A.HIGH VINTCT0  ;Interrupt HOB
181C   ED 47            LD      I.A             ;  ins I-Register

181E   ED 5E            IM      2               ;Interrupt-Mode 2
1820   FB               EI                      ;IFF1-Flag setzen

                        ;-----------------------------------------------

                        ;   Programm-Voraussetzungen

1821   3E 01            LD      A.00000001B     ;1. Ausgabe-Bitmuster
1823   0E 03            LD      C.32H           ;Zähler für Lauflicht

1825   C9               RET                     ;Ende der Routine

                    ;*****************************************************
```

```
                        ;*******************************************************

1826                    INTCTO: ;Interrupt-Service-Routine für CTC
                                ;-------------------------------
                                ;       Das Schieben des Bitmusters im Akkumula-
                                ;       tor erfolgt erst nach einer vorgegebenen
                                ;       Anzahl von Interrupts.

1826    F5                      PUSH    AF              ;(Akkumulator) verwahren

1827    0D                      DEC     C               ;(Interr.-Zähler) dekr.
1828    28 03                   JR      Z.SCHIEB        ;(interr.-Zähler) = 0 ?
182A    F1                      POP     AF              ;  Nein: (Akku) vom Stack
182B    18 06                   JR      ENDR            ;        und Ende

182D                    SCHIEB: ;       Der Interrupt-Zähler wird neu gesetzt.
                                ;       Erst nach der vorgegebenen Anzahl von In-
                                ;       terrupts wird das Bitmuster im Akkumula-
                                ;       tor mit einem 1-Bit über den PIO-Port B
                                ;       ausgegeben. Das 1-Bit wird im Akkumulator
                                ;       links im Kreis geschoben.

182D    0E 03                   LD      C.32H           ;Interrupt-Zähler setzen

182F    F1                      POP     AF              ;(Akku) vom Stack holen
1830    D3 81                   OUT     (PIODB).A       ;Bitmuster nach Port B
1832    07                      RLCA                    ;1-Bit links schieben

1833    FB              ENDR:   EI                      ;IFF1 setzen
1834    ED 4D                   RETI                    ;Ende der Routine

                        ;*******************************************************

                                ORG     1838H

                                ;Vektorfeld     Ablage-Speicherzelle für die An-
                                :----------     fangsadresse der Interrupt-Ser-
                                ;               vice-Routine

1838    26 18           VINTCTO:DW      INTCTO          :CTC-Interrupt

                        ;*******************************************************

                                END
```

50 32H
100 64H
150 96H
200 C8H
250 FAH

37 30 } 25 sec 80 x geschoben
37 55
0,3125 sec für Schritt
: 250 = 1.25 ms
/ 140 ≙ 8.9 usec

Channel Control-Word 87 pro Countdown

45 57 } 40 sec 8 x geschoben
46 37 5 sec per Schritt
: 250 = 20 usec per Interr.
: 140 = 143 usec pro Count down
x 256 = 559 ms / Clock
≙ 1.79 MHz Quarz

Channel Control Word
87

```
                        ;*******************************************************
                        ;*                Lauflicht mit Sekunden               *
                        ;*                ----------------------               *
                        ;*   Ein Lauflicht wird über Kanal B der PIO gesteuert. *
                        ;*    über Interrupt im Kanal A der PIO wird die Pro-   *
                        ;*     grammierung des CTC so ein- und ausgeschaltet.   *
                        ;*    daß wahlweise akustische Sekunden-Signale ausge-  *
                        ;*                geben werden oder nicht.             *
                        ;*                                                     *
                        ;*******************************************************

                               ;Definitionen

0040                    CTC0   EQU    40H     ;CTC Kanal 0

0080                    PIODA  EQU    80H     ;Daten   PIO-Port A
0081                    PIODB  EQU    81H     ;Daten   PIO-Port B
0082                    PIOBA  EQU    82H     ;Befehle PIO-Port A
0083                    PIOBB  EQU    83H     ;Befehle PIO-Port B

05DE                    TONE1K EQU    05DEH   ;1 kHz-Periode auf Lautsprecher

                               ORG    1800H

                        ;*******************************************************

1800  CD 17 18          ANFANG: CALL   INIT

1803                    BLINK: ;Hauptprogramm: Das Bitmuster im Akkumualtor mit
                               ;------------  einem 1-Bit wird über den Port B
                               ;              ausgegeben. Das 1-Bit wird im
                               ;              Akku nach links im Kreis gescho-
                               ;              ben. Nach Verzögerung erfolgt er-
                               ;              neute Ausgabe über Port B.

1803  D3 81                          OUT    (PIODB).A       ;Bitmuster nach Port B
1805  CD 0B 18                       CALL   WAIT            ;Verzögerung
1808  07                            RLCA                   ;1-Bit links schieben
1809  18 F8                         JR     BLINK           ;Nächstes Bitmuster anz.

                        ;*******************************************************

180B                    WAIT:  ;Unterprogramm: Warteschleife zur Einstellung der
                               ;------------  Geschwindigkeit des Lauflichts

180B  F5                      WAIT:  PUSH   AF              ;(Akkumulator) verwahren
180C  21 00 B0                       LD     HL.0B000H      ;Länge der Schleife
180F  11 01 00                       LD     DE.1

1812  19                      WAITL: ADD    HL.DE
1813  30 FD                         JR     NC.WAITL

1815  F1                            POP    AF              ;(Akkumulator) vom Stack
1816  C9                            RET                    ;Ende der Routine

                        ;*******************************************************
```

```
                        ;*****************************************************

1817                    INIT:   ;Unterprogramm  Programmierung der PIO-Ports A.B.
                                ;-------------  Programmierung des CTC-Kanals 0
                                ;               Für nachträgliche Programmierung
                                ;               des CTC-Kanals 1 ist Platz reser-
                                ;               viert.
                                ;               Interrupt-Programmierung der CPU.

                                ;    PIO-Port A
                                ;               Mode Control Word

1817    3E CF                   LD      A.0CFH          ;Betriebsart 3
1819    D3 82                   OUT     (PIOBA).A

                                ;               I/O Register Control Word

181B    3E 01                   LD      A.00000001B     ;Bit Nr.0 wird Eingang
181D    D3 82                   OUT     (PIOBA).A

                                ;               Interrupt Vector Word

181F    3E BE                   LD      A.LOW VINTPIO   ;LOB Ablage-Adresse
1821    D3 82                   OUT     (PIOBA).A

                                ;               Interrupt Control Word

1823    3E B7                   LD      A.10110111B     ;Bit Nr.0-3: Kennung
                                                        ;Bit Nr.4: Maske folgt
                                                        ;Bit Nr.5: L-H - Interr.
                                                        ;Bit Nr.6: (Don't care)
                                                        ;Bit Nr.7: Interrupt frei
1825    D3 82                   OUT     (PIOBA).A       ;  Control

                                ;               Mask Control Word

1827    3E FE                   LD      A.11111110B     ;Interr. über Bit Nr.0
1829    D3 82                   OUT     (PIOBA).A

                                ;-----------------------------------------------

                                ;    PIO-Port B
                                ;               Mode Control Word

182B    3E CF                   LD      A.0CFH          ;Betriebsart 3
182D    D3 83                   OUT     (PIOBB).A

                                ;               I/O Register Control Word

182F    3E 00                   LD      A.00000000B     ;Alle Anschlüsse Ausgänge
1831    D3 83                   OUT     (PIOBB).A

                                ;-----------------------------------------------
```

```
                                ;-----------------------------------------------

                                ;    CTC Kanal 0
                                ;              Channel Control Word

1833    3E 37                   LD      A.00110111B     ;Bit Nr.0: Kennung
                                                        ;Bit Nr.1: RESET
                                                        ;Bit Nr.2: Folgt Zeitk.
                                                        ;Bit Nr.3: Autom. Start
                                                        ;Bit Nr.4: (Don't Care)
                                                        ;Bit Nr.5: Teiler = 256
                                                        ;Bit Nr.6: Timer Mode
                                                        ;Bit Nr.7: Sperrt Interr.
1835    D3 40                   OUT     (CTC0).A

                                ;              Zeitkonstante

1837    3E 20                   LD      A.32
1839    D3 40                   OUT     (CTC0).A

                                ;              Interrupt Vector Word

183B    3E C0                   LD      A.LOW VINTCT0   ;Interrupt-Vektor LOB
183D    D3 40                   OUT     (CTC0).A

                                ;-----------------------------------------------

                                ;    Reserviert   für Programmierung CTC Kanal 1

183F    00 00 00 00 00 00 00 00 DB      0.0.0.0.0.0.0.0

                                ;-----------------------------------------------

                                ;    CPU-Programmierung

1847    3E 18                   LD      A.HIGH VINTPIO  ;Interrupt HOB
1849    ED 47                   LD      I.A             ;  ins I-Register

184B    ED 5E                   IM      2               ;Interrupt-Mode 2
184D    FB                      EI                      ;IFF1-Flag setzen

                                ;-----------------------------------------------

                                ;    Programm-Voraussetzungen

184E    3E 01                   LD      A.00000001B     ;1. Ausgabe-Bitmuster
1850    B7                      OR      A               ;Carry-Bit löschen
1851    0E 00                   LD      C.0             ;Zähler für Lauflicht

1853    DD 21 B4 18             LD      IX.START        ;Zeiger auf Flag-Feld

1857    DD 36 00 01             LD      (IX+0).1        ;START-Flag wird 1
185B    DD 36 01 DA             LD      (IX+1).218      ;Zähler f. 218 Interrupts
185F    00 00 00                DB      00.00.00        ;Reserviert für Befehl

1862    C9                      RET                     ;Ende der Routine
```

```
                        ;*****************************************************

1863                    INTPIO: ;Interrupt-Service-Routine für PIO
                        ;-------------------------------
                        ;       Je nach Wert des START-Flags wird der CTC
                        ;       zur Ausgabe von Interrupt-Anforderungen
                        ;       über den Kanal 0 im Sekunden-Takt pro-
                        ;       grammiert (LAUF) oder die Ausgabe ge-
                        ;       sperrt (STOP).

1863    08                      EX      AF.AF'          ;Zweiter Registersatz

1864    DD 7E 00                LD      A.(IX+0)        ;START-Flag holen
1867    A7                      AND     A               ;für Zero-Flag
1868    28 0A                   JR      Z.STOP          ;Wenn Z=0: Stop

                        ;-----------------------------------------------

186A    DD 36 00 00     LAUF:   LD      (IX+0).0        ;(START)=0 für nächsten
                                                        ;  PIO-Interrupt

                        ;    CTC Kanal 0
                        ;               Channel Control Word

186E    3E B1                   LD      A.10110001B     ;Bit Nr.1: Kein RESET
                                                        ;Bit Nr.2: Keine Zeitk.
                                                        ;Bit Nr.7: Interrupt frei
1870    D3 40                   OUT     (CTC0).A

1872    18 08                   JR      REND            ;Ende der Routine

                        ;-----------------------------------------------

1874    DD 36 00 01     STOP:   LD      (IX+0).1        ;(START)=1 für nächsten
                                                        ;  PIO-Interrupt

                        ;    CTC Kanal 0
                        ;               Channel Control Word

1878    3E 31                   LD      A.00110001B     ;Kein RESET
                                                        ;Bit Nr.2: Keine Zeitk.
                                                        ;Bit Nr.7: Interr. sperrt
187A    D3 40                   OUT     (CTC0).A

187C    08              REND:   EX      AF.AF'          ;1. Registersatz
187D    FB                      EI                      ;IFF1-Flag setzen
187E    ED 4D                   RETI                    ;Ende der Routine

                        ;*****************************************************
```

```
                        ;*****************************************************

1880                    INTCTO: ;Interrupt-Service-Routine für CTC
                                ;--------------------------------
                                ;       Eine Sekunden-Anzeige erfolgt erst nach
                                ;       218 (= DAH) Interrupts der CTC.

1880    08                      EX      AF.AF'          ;Zweiter Registersatz
1881    D9                      EXX

1882    DD 7E 01                LD      A.(IX+1)        ;(Interrupt-Zähler) holen
1885    3D                      DEC     A               ;  und dekrementieren
1886    28 05                   JR      Z.SEKUND        ;Bei 0 ist 1 Sekunde um
1888    DD 77 01                LD      (IX+1).A        ;Noch nicht:
188B    18 0A                   JR      ENDR            ;  Ende der Routine

188D                    SEKUND: ;       Erst nach 218 Interrupt ist ein Sekunde
                                ;       abgelaufen. Der Interrupt-Zähler wird er-
                                ;       neut auf 218 gesetzt.
                                ;       Die volle Sekunde wird akustisch gemeldet

188D    DD 36 01 DA             LD      (IX+1).218      ;Interrupt-Zähler setzen

1891    21 28 00                LD      HL.0028H        ;Anzahl 1-kHz-Perioden
1894    CD DE 05                CALL    TONE1K          ;1-kHz-Ton

1897    08              ENDR:   EX      AF.AF'          ;Erster Registersatz
1898    D9                      EXX
1899    FB                      EI                      ;IFF1 setzen
189A    ED 4D                   RETI                    ;Ende der Routine

                        ;*****************************************************

                                ;Reserviert
                                ;----------     für Interrupt-Service-Routine
                                ;               CTC Kanal 1

189C    00 00 00 00 00 00 00 00 DB      0.0.0.0.0.0.0.0
18A4    00 00 00 00 00 00 00 00 DB      0.0.0.0.0.0.0.0
18AC    00 00 00 00 00 00 00 00 DB      0.0.0.0.0.0.0.0

                        ;*****************************************************
```

```
                        ;*******************************************************
                        START:  ;Flag-Feld      Wird vom Inhalt des IX-Registers
                                ;---------      adressiert

18B4                            DB      1               ;START-Flag.      Anf:  1
18B5    DA                      DB      218             ;Interrupt-Zähler Anf:218
18B6    00                      DB      00              ;Reserviert für Flag

                        ;*******************************************************

                                ORG     18BEH

                                ;Vektorfeld     Ablage-Speicherzellen fÜr die An-
                                ;----------     fangsadressen der Interrupt-Ser-
                                ;               vice-Routinen

18BE    63 18           VINTPIO:DW      INTPIO          ;PIO-Interrupt
18C0    80 18           VINTCTO:DW      INTCTO          ;CTC-Interrupt
18C2    00 00                   DB      00.00           ;Reserve

                        ;*******************************************************

                                END
```

```
                        ;*******************************************************
                        ;*        Lauflicht - Ergänzungen - Änderungen         *
                        ;*        ------------------------------------         *
                        ;*    Die im folgenden aufgeführten Befehle können in  *
                        ;*      das Programm für  Lauflicht mit Sekunden an    *
                        ;*   den dafür bereits  vorgesehenen Stellen eingefügt *
                        ;*   werden. Sie bewirken eine Interrupt-Steuerung des *
                        ;*      Lauflichts über Impule vom Kanal 1 des CTC.    *
                        ;*******************************************************

0041                    CTC1    EQU     41H     ;CTC Kanal 1
0081                    PIODB   EQU     81H     ;Daten   PIO-Port B
1817                    INIT    EQU     1817H

                        ;*******************************************************

                                ORG     1803H

1803                    BLINK:  ;Hauptprogramm: Die CPU wartet im HALT-Zustand
                                ;-------------  auf Lauflicht- und Sekunden-In-
                                ;               terrupts.

1803    76                      HALT                    ;Auf Interrupt warten
1804    18 FD                   JR      BLINK           ;Nächster Interrupt

                        ;*******************************************************

                                ORG     183FH

                                ;    CTC Kanal 1
                                ;               Channel Control Word

183F    3E B7                   LD      A.10110111B     ;Bit Nr.0: Kennung
                                                        ;Bit Nr.1: RESET
                                                        ;Bit Nr.2: Folgt Zeitk.
                                                        ;Bit Nr.3: Autom. Start
                                                        ;Bit Nr.4: (Don't Care)
                                                        ;Bit Nr.5: Teiler = 256
                                                        ;Bit Nr.6: Timer Mode
                                                        ;Bit Nr.7: Interrupt frei
1841    D3 41                   OUT     (CTC1).A

                                ;               Zeitkonstante

1843    3E FF                   LD      A.0FFH
1845    D3 41                   OUT     (CTC1).A

                        ;*******************************************************

                                ORG     185FH

                                ;    Programm-Voraussetzungen

185F    DD 36 02 03             LD      (IX+2).3        ;Zähler für 3 Interrupts

                        ;*******************************************************
```

```
                        ;*****************************************************

                                ORG     189CH

189C                    INTCT1: ;Interrupt-Service-Routine für CTC Kanal 1
                                ;---------------------------------------

                                ;       Das Schieben des Bitmusters im Akkumula-
                                ;       tor erfolgt erst nach einer vorgegebenen
                                ;       Anzahl von Interrupts.

189C    F5                      PUSH    AF              ;(Akkumulator) verwahren

189D    DD 7E 02                LD      A.(IX+2)        ;(Interrupt-Zähler) holen
18A0    3D                      DEC     A               ;  und dekrementieren
18A1    28 06                   JR      Z.SCHIEB        ;(Zähler) = 0 ?
18A3    DD 77 02                LD      (IX+2).A        ;Noch nicht:
18A6    F1                      POP     AF              ;  (Akku) vom Stack holen
18A7    18 08                   JR      ISREND          ;  Ende der Routine


18A9                    SCHIEB: ;       Der Interrupt-Zähler wird neu gesetzt.
                                ;       Erst nach der vorgegebenen Anzahl von In-
                                ;       terrupts wird das Bitmuster im Akkumula-
                                ;       tor mit einem 1-Bit über den Port B aus-
                                ;       gegeben. Das 1-Bit wird im Akku nach
                                ;       links im Kreis geschoben.

18A9    DD 36 02 03             LD      (IX+2).3        ;Interrupt-Zähler setzen

18AD    F1                      POP     AF              ;(Akku) vom Stack holen
18AE    D3 81                   OUT     (PIODB).A       ;Bitmuster nach Port B
18B0    07                      RLCA                    ;1-Bit links schieben

18B1    FB              ISREND: EI                      ;IFF1 setzen
18B2    ED 4D                   RETI                    ;Ende der Routine

                        ;*****************************************************

                                ORG     18B6H

                                ;Flag-Feld      Wird von (IX) adressiert

18B6    03                      DB      3               ;Interrupt-Zähler Anf:  3

                        ;*****************************************************

                                ORG     18C2H

                                ;Vektorfeld     Ablage für die Anf.-Adr. der In-
                                                terrupt-Service-Routine INTCT1

18C2    9C 18           VINTCT1:DW      INTCT1          ;CTC-Interrupt Kanal 1

                        ;*****************************************************
```

```
                ;*******************************************************
                ;*                                                     *
                ;*         Serielle Ausgabe von Daten-Bytes            *
                ;*          über Timer-Interrupt gesteuert             *
                ;*               mit optischer Anzeige                 *
                ;*                                                     *
                ;*******************************************************

                ;Definitionen:

                PIODA   EQU     80H     ;Daten   PIO-Port A
                PIODB   EQU     81H     ;Daten   PIO-Port B

                PIOBA   EQU     82H     ;Befehle PIO-Port A
                PIOBB   EQU     83H     ;Befehle PIO-Port B

                CTC0    EQU     40H     ;CTC Kanal 0
                CTC1    EQU     41H     ;CTC Kanal 1

                ;-------------------------------------------------------

                        ;Die CTC-Zeitkonstanten werden für einen Bit-
                        ;Abstand von etwa 0.5 s eingestellt

                TC0     EQU     14      ;CTC Kanal 0
                TC1     EQU     00      ;CTC Kanal 1

                ANFAD   EQU     ANFANG  ;Anfangs-Adresse für Ausgabe

                        ORG     1800H

                ;*******************************************************

1800  CD 06 18        ANFANG: CALL    INIT

                      AUSGAB: ;Hauptprogramm:
                              ;-------------

                              ;       Die CPU wartet im HALT-Zustand auf CTC-
                              ;       Interrupts. Diese verursachen die seri-
                              ;       elle Ausgabe jeweils eines Bits.

1803  76                      HALT                    ;Auf Interrupt warten
1804  18 FD                   JR      AUSGAB          ;Nächster Interrupt

                ;*******************************************************
```

```
                        ;*****************************************************

                        INIT:   ;Unterprogramm: Programmierung der Interface-
                                ;------------   Bausteine

                                ;     PIO-Port A

                                ;        Das seriell ausgegebene Bitmuster wird
                                ;        von links her in die Port-Anzeige ein-
                                ;        geschoben und als Leuchtdioden-Muster
                                ;        angezeigt.

                                ;             Mode Control Word

1806  3E CF                             LD      A.0CFH          ;Betriebsart 3
1808  D3 82                             OUT     (PIOBA).A

                                :             I/O Register Control Wort

180A  3E 00                             LD      A.00000000B     ;Alle Anschlüsse Ausgänge
180C  D3 82                             OUT     (PIOBA).A

                                ;-----------------------------------------------

                                ;     PIO-Port B

                                ;        Der serielle Datenstrom wird über Bit
                                ;        Nr.3 des Datenports B ausgegeben. Das
                                ;        Bit Nr.7 wird als Eingang programmiert
                                ;        für das CTS-Signal.

                                ;             Mode Control Word

180E  3E CF                             LD      A.0CFH          ;Betriebsart 3
1810  D3 83                             OUT     (PIOBB).A

                                :             I/O Register Control Wort

1812  3E 80                             LD      A.10000000B     ;Bit Nr.7 wird Eingang
1814  D3 83                             OUT     (PIOBB).A

                        ;-----------------------------------------------------
```

```
                        ;-------------------------------------------------

                        ;     CTC Kanal 0

                        ;       Kanal 0 wird von Kanal 1 getriggert.
                        ;       Die Impulse des Kanals 0 im CTC unter-
                        ;       brechen den CPU-Ablauf zur seriellen
                        ;       Ausgabe eines Bits.

                        ;             Channel Control Word

1816  3E FF            LD      A.11111111B     ;Bit Nr.0: Kennung
                                                ;Bit Nr.1: Reset
                                                ;Bit Nr.2: Folgt Zeitk.
                                                ;Bit Nr.3: CLK/TRG trigg.
                                                ;Bit Nr.4: L-H - Flanke
                                                ;Bit Nr.5: Don/t care
                                                ;Bit Nr.6: Counter Mode
                                                ;Bit Nr.7: Interr. frei
1818  D3 40             OUT     (CTC0).A

                        ;             Time Constant Word

181A  3E 0E             LD      A.TC0           ;Zeitkonstante
181C  D3 40             OUT     (CTC0).A

                        ;             Interrupt Vector Word

181E  3E A8             LD      A.LOW INTVEK    ;Interrupt Vektor LOB
1820  D3 40             OUT     (CTC0).A

                        ;-------------------------------------------------

                        ;     CTC Kanal 1

                        ;       Kanal 1 wird von dem durch 256 geteil-
                        ;       ten System-Takt getriggert.

                        ;             Channel Control Word

1822  3E 37             LD      A.00110111B     ;Bit Nr.0: Kennung
                                                ;Bit Nr.1: Reset
                                                ;Bit Nr.2: Folgt Zeitk.
                                                ;Bit Nr.3: Autom. trigg.
                                                ;Bit Nr.4: L-H - Flanke
                                                ;Bit Nr.5: Teiler = 256
                                                ;Bit Nr.6: Timer Mode
                                                ;Bit Nr.7: Kein Interrupt
1824  D3 41             OUT     (CTC1).A

                        ;             Time Constant Word

1826  3E 00             LD      A.TC1           ;Zeitkonstante
1828  D3 41             OUT     (CTC1).A

                        ;-------------------------------------------------
```

```
                        ;-------------------------------------------------

                                ;   CPU-Programmierung

182A  3E 18                     LD      A.HIGH INTVEK   ;Interrupt-Vektor HOB
182C  ED 47                     LD      I.A             ;  ins I-Register

182E  ED 5E                     IM      2               ;Interrupt-Mode 2
1830  FB                        EI                      ;IFF1 setzen

                      ;-------------------------------------------------

                                ;   Programm-Voraussetzungen

                                   ;Die Parameter für die Interrupt-Servi-
                                   ;ce-Routine werden dem zweiten Regi-
                                   ;ster-Satz der CPU übergeben.

1831  D9                        EXX                     ;2. Register-Satz

1832  21 00 18                  LD      HL.ANFAD        ;Anfangs-Adresse
1835  06 0B                     LD      B.11            ;11 serielle Bits
1837  0E 00                     LD      C.0             ;Noch kein CTS
1839  11 00 00                  LD      DE.0            ;Ablagen löschen

183C  D9                        EXX                     ;1. Register-Satz

                                   ;Wenn nicht gesendet wird. steht auf
                                   ;der Ausgangsleitung im Kanal B ein 1-
                                   ;Signal. Das Start-Bit wird mit dem
                                   ;Wert 1 angezeigt.

183D  3E 08                     LD      A.00001000B     ;1-Signal
183F  D3 81                     OUT     (PIODB).A

                      ;-------------------------------------------------

1841  C9                        RET                     ;Ende der Routine

                      ;*************************************************
```

```
                        ;******************************************************
                        ;
                        SEROUT: ;Interrupt-Service-Routine
                                ;------------------------

                                ;       Die SEROUT-Routine wird bei jedem Takt
                                ;       des Timers zum Aussenden eines Bits an-
                                ;       gesprungen.

                                ;       Die Routinen-Parameter stehen im 2. Regi-
                                ;       ster-Satz der CPU.

                                ;       CTS setzt ein Flag. Wenn weder CTS em-
                                ;       pfangen noch das CTS-Flag gesetzt ist,
                                ;       wird die Routine sofort verlassen.

                                ;       Der Bit-Zähler ist zunächst auf 11 ge-
                                ;       setzt. Sein Inhalt wird nach CTS dekre-
                                ;       mentiert. - Das erste Bit wird als Start-
                                ;       0-Bit ausgesendet. Es folgen 8 Daten-Bits
                                ;       und zwei Stop-1-Bits. Nach 2. Stop-Bit
                                ;       wird das CTS-Flag gelöscht. Zähler = 11.

1842  D9                        EXX                     ;2. Register-Satz

1843  DB 81                     IN      A.(PIODB)       ;CTS abfragen
1845  E6 80                     AND     10000000B
1847  20 06                     JR      NZ.SET          ;Ja: CTS-Flag setzen

1849  79                        LD      A.C             ;CTS-Flag abfragen
184A  A7                        AND     A
184B  28 2F                     JR      Z.RETURN        ;Kein CTS
184D  18 02                     JR      DECR

                        SET:    ;         Das CTS-Flag wird gesetzt

184F  0E FF                     LD      C.0FFH

                        DECR:   ;         Bit-Zähler dekrementieren und nach
                                ;         Start-Bit fragen.

1851  05                        DEC     B               ;Zähler - 1
1852  78                        LD      A.B
1853  FE 0A                     CP      10              ;Start-Bit?
1855  20 0A                     JR      NZ.DATEN        ;  Nein: Daten-Bit

                                ;         Start-Bit. Byte nach D-Register holen

1857  11 00 00                  LD      DE.0            ;Ablage löschen

185A  CD 80 18                  CALL    STARTB          ;Start-Bit senden
185D  56                        LD      D.(HL)          ;Byte nach D-Reg.
185E  23                        INC     HL              ;(Pointer) erhöhen

185F  18 1B                     JR      RETURN
```

```
                        DATEN:  ;       Bit-Zähler fragen. ob bereits 8 Daten-
                                ;       Bits gesendet wurden.

1861  78                            LD      A.B
1862  FE 02                         CP      2               ;Noch Daten-Bits?
1864  38 0A                         JR      C.STOP          ;  Nein: Stop-Bits

                                ;       Daten-Bit Nr.0 im D-Register wird
                                ;       Carry-Bit. Daten-Byte im D-Register
                                ;       wird rechts rotiert.

1866  CB 0A                         RRC     D

                                ;       Je nach Wert des Carry-Bits wird ein 0-
                                ;       oder ein 1-Bit gesendet.

1868  DC 87 18                      CALL    C.EINS          ;1-Bit senden
186B  D4 96 18                      CALL    NC.NULL         ;0-Bit senden

186E  18 0C                         JR      RETURN

                        STOP:   ;       Aussenden eines Stop-1-Bits.

                                ;       Nach dem 2. Stop-Bit wird das CTS-Flag
                                ;       gelöscht und der Bit-Zähler auf 11 ge-
                                ;       setzt.

1870  78                            LD      A.B
1871  FE 01                         CP      1               ;1. Stop-Bit?
1873  28 04                         JR      Z.ERST          ;  Ja
                                                            ;  Nein:
1875  0E 00                         LD      C.0             ;CTS-Flag löschen
1877  06 0B                         LD      B.11            ;(Zähler) = 11

1879  CD A3 18               ERST:  CALL    STOPB           ;Stop-Bit senden

                        RETURN: ;       Auf ersten Register-Satz umschalten.
                                ;       Ende der Interrupt-Routine.

187C  D9                            EXX                     ;1. Register-Satz
187D  FB                            EI                      ;IFF1-Flag setzen
187E  ED 4D                         RETI                    ;Ende der Routine

                        ;*****************************************************

                        STARTB: ;Unterprogramm: Aussenden eines Start-Bits
                                ;-------------

                                ;       Die Ausgangs-Leitung im Port B wird auf
                                ;       0 geschaltet.

1880  3E 00                         LD      A.00000000B
1882  D3 81                         OUT     (PIODB).A
```

```
                        ;       Die Byte-Anzeige im Port A wird auf 0
                        ;       geschaltet.

1884  D3 80             OUT     (PIODA).A

1886  C9                RET                     ;Ende der Routine

                    ;*****************************************************

                    EINS:   ;Unterprogramm: Ausgabe eines 1-Daten-Bits
                            ;-------------

                            ;       Das Muster der bereits ausgesendeten
                            ;       Daten-Bits wird im E-Register verwahrt.
                            ;       Das jeweils folgende Bit wird über Port
                            ;       A von links nach rechts eingeschoben
                            ;       und angezeigt. - Das Bit wird als Nr.3
                            ;       über Port B ausgegeben.

1887  F5                PUSH    AF

1888  3E 08             LD      A.00001000B     ;Ausgabe Port B
188A  D3 81             OUT     (PIODB).A

                            ;       Das Muster der bereits ausgesendeten
                            ;       Daten-Bits wird im E-Register verwahrt.
                            ;       Das jeweils folgende Bit wird über Port
                            ;       A von links nach rechts eingeschoben
                            ;       und angezeigt.

188C  7B                LD      A.E             ;Muster holen
188D  CB 3F             SRL     A               ; und schieben
188F  F6 80             OR      10000000B       ;Bit dazu
1891  5F                LD      E.A             ;Muster verwahren
1892  D3 80             OUT     (PIODA).A       ; und ausgeben

1894  F1                POP     AF
1895  C9                RET                     ;Ende der Routine

                    ;*****************************************************

                    NULL:   ;Unterprogramm: Ausgabe eines 0-Daten-Bits
                            ;-------------
                            ;       Das Muster der bereits ausgesendeten
                            ;       Daten-Bits wird im E-Register verwahrt.
                            ;       Das jeweils folgende Bit wird über Port
                            ;       A von links nach rechts eingeschoben
                            ;       und angezeigt. - Das Bit als als Nr.3
                            ;       über Port B ausgegeben.

1896  F5                PUSH    AF

1897  3E 00             LD      A.00000000B     ;Ausgabe Port B
1899  D3 81             OUT     (PIODB).A
```

```
                        ;       Das Muster der bereits ausgesendeten
                        ;       Daten-Bits wird im E-Register verwahrt.
                        ;       Das jeweils folgende Bit wird über Port
                        ;       A von links nach rechts eingeschoben
                        ;       und angezeigt.

189B  7B                LD      A.E             ;Muster holen
189C  CB 3F             SRL     A               ; und schieben
189E  5F                LD      E.A             ;Muster verwahren
189F  D3 80             OUT     (PIODA).A       ; und ausgeben

18A1  F1                POP     AF
18A2  C9                RET                     ;Ende der Routine

                ;*****************************************************

                STOPB:  ;Unterprogramm: Ausgabe eines Stop-1-Bits
                        ;-------------
                        ;       Ein Stop-Bit wird über Port B auf die
                        ;       Ausgangs-Leitung geschaltet.

18A3  3E 08             LD      A.00001000B
18A5  D3 81             OUT     (PIODB).A

18A7  C9                RET                     ;Ende der Routine

                ;*****************************************************

                        ORG     (($-1) OR 00000111B)+1

                INTVEK: ;Vektorfeld: Ablage des Interrupt-Vektors
                        ;----------

18A8  42 18             DW      SEROUT

                ;*****************************************************

                        END
```

S.69
PER14S69/EDI

```
                        ;*************************************************
                        ;*                                               *
                        ;*          Binärer Zähler über Port C           *
                        ;*                                               *
                        ;*************************************************

                        ;Definitionen:

                        STW     EQU     0C3H            ;STEUERWORT AN 8255
                        PORTC   EQU     0C2H            ;PORTC

                        ;*************************************************

                        ;Hauptprogramm

                                ORG     1800H

1800  CD 40 18        START:  CALL    INIT            ;Initialisierung 8255
1803  3E 00                  LD      A,00H           ;Akku mit 00 laden
1805  D3 C2           LOOP0:  OUT     (PORTC),A       ;Ausgabe an Port C
1807  CD 50 18                CALL    ZEIT            ;Unterprogramm ZEIT
180A  3C                      INC     A               ;Zähler inkrementieren
180B  20 F8                   JR      NZ,LOOP0        ;Schleife 0
180D  76                      HALT                    ;Ende des Programms

                        ;*************************************************

                        ;Unterprogramm INIT

                                ORG     1840H

1840  3E 90              INIT:   LD      A,90H           ;Steuerwort laden
1842  D3 C3                    OUT     (STW),A         ;Steuerwort einschr.
1844  C9                      RET                     ;zurück zum Hauptprg.

                        ;*************************************************

                        ;Unterprogramm ZEIT

                                ORG     1850H

1850  E5                ZEIT:   PUSH    HL              ;HL retten
1851  F5                      PUSH    AF              ;AF retten
1852  26 FF                   LD      H,0FFH          ;Zähler 1 laden
1854  2E FF              LOOP1:  LD      L,0FFH          ;Zähler 2 laden
1856  2D                LOOP2:  DEC     L               ;Zähler 1 dekrement
1857  20 FD                   JR      NZ,LOOP2        ;Sprung bei Z=00
1859  25                      DEC     H               ;Zähler 2 dekrement
185A  20 F8                   JR      NZ,LOOP1        ;Sprung bei Z=00
185C  F1                      POP     AF              ;AF zurückholen
185D  E1                      POP     HL              ;HL zurückholen
185E  C9                      RET                     ;zurück zu Hauptprg.

                        ;*************************************************

                                END
```

```
                        ;*****************************************************
                        ;*                                                   *
                        ;* Ein Leuchtpunkt wird als Lauflicht über den PORTA *
                        ;* des 8255 geschoben. Über PORTC des 8255 wird die  *
                        ;* Konstante für die Zeitverzögerung eingelesen.     *
                        ;*                                                   *
                        ;*****************************************************

                        ;Definitionen:

                        PORTA   EQU     0C0H            ;Portadresse für PORT A
                        PORTC   EQU     0C2H            ;Portadresse für PORT C
                        STW     EQU     0C3H            ;Steuerwort-Adresse
                        STW1    EQU     10001011B       ;Steuerwort

                                ORG     1800H
                        ;*****************************************************
1800  CD 17 18         START:  CALL    INIT            ;Anfangsbedingungen

1803  37               LOOP:   SCF                     ;Carry-Bit = 1
1804  3E 00                     LD      A,00000000B
1806  D3 C0            LOOP1:  OUT     (PORTA),A       ;Ausgabe an PORT A

1808  F5                        PUSH    AF
1809  CD 29 18                  CALL    ZEIT1           ;Zeitverzögerung 1
180C  F1                        POP     AF

180D  17                        RLA                     ;Akku links schieben
180E  30 F6                     JR      NC,LOOP1        ;Schleife 1
1810  D3 C0                     OUT     (PORTA),A       ;Ausgabe an PORT A
1812  CD 1C 18                  CALL    ZEIT2           ;Zeitverzögerung 2
1815  18 EC                     JR      LOOP            ;zurück zum Anfang
                        ;*****************************************************
1817  3E 8B             INIT:   LD      A,STW1          ;Steuerwort in das
1819  D3 C3                     OUT     (STW),A         ; Steuerwortregister
181B  C9                        RET                     ;Unterprogramm beenden
                        ;*****************************************************
181C  DB C2             ZEIT2:  IN      A,(PORTC)       ;PORTC einlesen
181E  6F                        LD      L,A             ;Zähler1 laden
181F  2C                        INC     L
1820  E5                 LOOP2:  PUSH    HL
1821  CD 29 18                  CALL    ZEIT1           ;Zeitverzögerung 1
1824  E1                        POP     HL
1825  2D                        DEC     L               ;Dekrementieren
1826  20 F8                     JR      NZ,LOOP2        ;Schleife 2
1828  C9                        RET                     ;Ende Unterprogramm
                        ;*****************************************************
1829  26 0F             ZEIT1:  LD      H,0FH           ;Zähler laden
182B  2E F0             LOOP4:  LD      L,0F0H          ;Zähler laden
182D  2D                LOOP3:  DEC     L               ;Zähler dekrementieren
182E  20 FD                     JR      NZ,LOOP3        ;Zähler leer?
1830  25                        DEC     H               ;Zähler dekrementieren
1831  20 F8                     JR      NZ,LOOP4        ;Zähler leer?
1833  C9                        RET                     ;Ende Unterprogramm
                        ;*****************************************************
```

```
                        ;************************************************
                        ;*        Interrupt-Zähler mit Anzeige          *
                        ;************************************************

                                ORG     1800H

                        ;************************************************

1800  21 00 19    START:  LD      HL,1900H        ;Anfangsadresse der
1803  22 EE 1F            LD      (1FEEH),HL      ;Interrupt-Routine

1806  1E 00              LD      E,00            ;Zähler Anfangsstand
1808  FB                EI                      ;Interrupt freigeben

1809  3E 90              LD      A,090H          ;Steuerwort nach Akku
180B  D3 03              OUT     (03),A          ;Seuerwortreg.8255

180D  3E 41              LD      A,01000001B     ;Bitmuster für Anzeige
180F  D3 02              OUT     (02),A          ; an Port C
1811  7B          ANZEI:  LD      A,E             ;Zählerstand in den Akku
1812  CD 89 06            CALL    0689H           ;Umcodierungsprogramm
1815  D3 01              OUT     (01),A          ;Segmente ansteuern
1817  C3 11 18            JP      ANZEI

                        ;************************************************

                                ORG     1900H

1900  F3          INTER:  DI                      ;Interrupt sperren
1901  CD 50 19            CALL    ZEIT            ;Zeitverzögerung
1904  1C                INC     E               ;Zählerstand erhöhen
1905  3E 0A              LD      A,0AH           ;Vergleichszahl laden,
1907  BB                CP      E               ;  mit E vergleichen
1908  20 02              JR      NZ,LOOP         ;Zähler löschen wenn er
                                                 ;größer ist als 09
190A  1E 00              LD      E,00            ;Zähler löschen
190C  FB          LOOP:   EI                      ;Interrupt freigeben
190D  ED 4D              RETI                    ;Ende der Routine

                        ;************************************************

                                ORG     1950H

1950  E5          ZEIT:   PUSH    HL              ;HL auf Stack retten

1951  26 FF              LD      H,0FFH          ;Zähler laden
1953  2E AF        LOOP2:  LD      L,0AFH          ;Zähler laden
1955  2D          LOOP1:  DEC     L               ;Zähler abwärtszählen
1956  20 FD              JR      NZ,(LOOP1)      ;  bis (Zähler) = 00
1958  25                DEC     H               ;Zähler abwärtszählen
1959  20 F8              JR      NZ,(LOOP2)      ;  bis (Zähler) = 00

195B  E1                POP     HL              ;HL vom Stack zurück
195C  C9                RET                     ;Ende Unterprogramm

                        ;************************************************
```

PER14 S80/EDI:3 (ANZ=2)
S80 L70

```
                        ;***********************************************
                        ;*                                             *
                        ;*  Azeigeprogramm mit dem 8255 auf der Micro- *
                        ;*  Professor-Platine.                         *
                        ;*                                             *
                        ;***********************************************

                        ;Definitionen:

                        PORTB   EQU     01H             ;Portadresse für PORTB
                        PORTC   EQU     02H             ;Portadresse für PORTC
                        STW     EQU     03H             ;Steuerwortadresse
                        STW1    EQU     10010000B       ;Steuerwort
                                                          90H
                                ORG     1800H

                        ;***************************************************

1800  F5                ANZEI:  PUSH    AF              ;Alle Register retten
1801  C5                        PUSH    BC
1802  E5                        PUSH    HL

1803  CD 1D 18                  CALL    INIT            ;8255 initialisieren

1806  21 00 19                  LD      HL,1900H        ;Anfangsadresse des An-
                                                        ;  zeigepuffers laden
1809  1E 06                    LD      E,06H           ;Zähler laden
180B  AF                        XOR     A               ;Anfangsbedingung für
180C  3C                        INC     A               ;  das Schieberegister

180D  CD 26 18          LOOP:   CALL    STELLE          ;Anzeigestelle aktiv  mit alten Daten
1810  CD 2E 18                  CALL    DATEN           ;Daten an diese Stelle  neue Daten.

1813  CD 35 18                  CALL    ZEIT            ;Zeitverzögerung

1816  1D                        DEC     E               ;Zähler decrementieren
1817  20 F4                     JR      NZ,LOOP         ;Schleife, solange Null-
                                                        ;  flag nicht aktiv

1819  E1                        POP     HL              ;Alle Register vom Stack
181A  C1                        POP     BC              ;  zurück holen
181B  F1                        POP     AF

181C  C9                        RET                     ;Ende des Unterprogramms

                        ;***************************************************

181D  3E 90             INIT:   LD      A,STW1          ;Steuerwort an Steuer-
181F  D3 03                     OUT     (STW),A         ;  wortregister 8255
                                                          40H
1821  3E 40                     LD      A,01000000B     ;BREAK inaktivieren
1823  D3 02                     OUT     (PORTC),A

1825  C9                          RET                   ;Ende des Unterprogramms
                        ;***************************************************
```

* besser zuerst auf schwarz, dann call Helle, dann call neue Daten

```
                        ;*****************************************************
                                                4014
1826  F6 40             STELLE: OR      01000000B       ;MASKE
1828  D3 02                     OUT     (PORTC),A       ;Stelle aktivieren
182A  EE 40                     XOR     01000000B       ;Demaskieren
182C  17                        RLA             40H     ;Nächste Stelle ein-
                                                        ;  stellen
182D  C9                        RET                     ;Ende des Unterprogramms

                        ;*****************************************************

182E  F5                DATEN:  PUSH    AF              ;Akkuinhalt retten

182F  7E                        LD      A,(HL)          ;(Anzeigepuffer) nach
1830  D3 01                     OUT     (PORTB),A       ;  Port B

1832  23                        INC     HL              ;nächste Stelle des
                                                        ;Anzeigepuffers

1833  F1                        POP     AF              ;Alter (Akkumulator)
1834  C9                        RET                     ;Ende des Unterprogramms

                        ;*****************************************************

1835  06 FF             ZEIT:   LD      B,255           ;Zähler laden

1837  05                LOOP1   DEC     B               ;Zähler bis Null
1838  20 FD                     JR      NZ,LOOP1        ;  dekrementieren

183A  C9                        RET                     ;Ende des Unterprogramms

183B                    ;*****************************************************

                                ~~END~~
```

Start:

1850 CD 0018 Call 1800H

1853 C3 5018 JP 1850H

1856

1900 Defb 0AFH ; 6 nach rechts

1901 " 0AEH ; 5 ⟶ ä f ö cd 1010|1110 A E H

" 036H ; 4

" 0BAH ; 3

" 093H ; 2

" 0301H ; 1 nach links

S. III - 58 MPF-Handbuch

PER14 S84/ED1:3 (ANZ=1)
S84 L72

```
                        ;*****************************************************
                        ;*                                                   *
                        ;*      Bit-Manipulationen über Port C des 8255      *
                        ;*                                                   *
                        ;*****************************************************

                        ;Definitionen

                        STW1    EQU     0C3H            ;Steuerwort-Register

                                ORG     1800H

                        ;*****************************************************

1800  3E 80               START:  LD      A,80H           ;Steuerwort: Alle Ports
1802  D3 C3                       OUT     (STW1),A        ;  Ausgabe

1804  3E 0F               LOOP1:  LD      A,0FH           ;Steuerwort für Bit-
1806  D3 C3               LOOP2:  OUT     (STW1),A        ;  manipulationen

1808  CD 70 18                   CALL    ZEIT            ;Warteschleife

180B  3D                       DEC     A               ;Akku dekrementieren
180C  20 F8                    JR      NZ,LOOP2        ;  und weiter ausgeben

180E  D3 C3                    OUT     (STW1),A        ;  Bit 0 auf 0
1810  C3 04 18                 JP      LOOP1           ;erneut zum Anfang

                        ;*****************************************************

                                ORG     1870H

1870  E5                 ZEIT:   PUSH    HL              ;(HL) retten

1871  26 0F                     LD      H,0FH           ;Zähler laden
1873  2E FF               LOOP4:  LD      L,0FFH          ;Zähler laden

1875  2D                 LOOP3:  DEC     L               ;Zähler dekrementieren
1876  20 FD                     JR      NZ,(LOOP3)      ;  bis (Zähler) = 00
1878  25                       DEC     H               ;Zähler dekrementieren
1879  20 F8                     JR      NZ,(LOOP4)      ;  bis (Zähler) = 00

187B  E1                       POP     HL              ;HL zurückholen
187C  C9                       RET                     ;Ende des Zeitprogramms

                        ;*****************************************************
```

CALL ZEIT

1816

END

PERI4 S 90 /ED1 : 3 (ANZ=2)
S 90 L 73

```
                ;*****************************************************
                ;*                                                   *
                ;*       Testprogramm für 8255 Betriebsart 1         *
                ;*    Eingabe Port A. Bits C4 und C5 sind Signale.   *
                ;*                                                   *
                ;*****************************************************

                ;Definitionen:

                PORTA   EQU     0C0H            ;Port A 8255 Peripherie
                PORTC   EQU     0C2H            ;Port C 8255 Peripherie
                STWR    EQU     0C3H            ;Steuerwortregister 8255
                STW     EQU     10110111B B7H   ;Steuerwort für 8255
                STW1    EQU     01111001B 79H   ;Steuerwort für INTE A

                ;*****************************************************

                        ORG     1800H

                ;Hauptprogramm

1800 3E B7      START:  LD      A,STW           ;Steuerwort ins
1802 D3 C3              OUT     (STWR),A        ;  Steuerwortregister

1804 21 00 19           LD      HL,~~1900H~~ INTERR ;Anfangsadresse der
1807 22 EE 1F           LD      (1FEEH),HL      ;Interrupt-Routine

180A 3E 79              LD      A,STW1          ;Steuerwort für INTE A-
180C D3 C3              OUT     (STWR),A        ;Freigabe ausgeben

180E FB                 EI                      ;Interrupt freigeben

180F 00         WARTE:  NOP                     ;Auf einen Interrupt
1810 C3 0F 18           JP      WARTE           ;warten

                ;*****************************************************
```

hier geht $\overline{INT}$ CPU bereits auf Ø
siehe H 78.1

Nach INI: IBF_B +1 IBFA +1 Rest Ø

```
                        ;*****************************************************

                                ORG     1900H

                        ;Interrupt-Service-Routine

1900  F3                INTERR: DI      ;Interrupt sperren
1901  DB C0                    IN      A,(PORTA)       ;Port A einlesen
1903  47                      LD      B,A             ;(Akku) nach B
1904  3E 7F                   LD      A,7FH           ;Bitstruktur für C7
                                                      ;   des 8255 setzen
1906  D3 C3                   OUT     (STWR),A        ;Ausgabe an Steuerwort-
                                                      ;   Register
1908  CD 12 19                CALL    ZEIT            ;Zeitverzögerung
190B  3E 7E                   LD      A,7EH           ;Bitstruktur für C7
190D  D3 C3                   OUT     (STWR),A        ;des 8255 rücksetzen
190F  FB                      EI                      ;Interrupt freigeben
1910  ED 4D                   RETI                    ;Ende von INTERR
                        ;*****************************************************

                        ;Unterprogrammm Zeit

1912  E5                ZEIT:   PUSH    HL              ;HL auf Stack retten
1913  26 FF                     LD      H,0FFH          ;Zähler laden
1915  2E AF              LOOP4:  LD      L,0AFH          ;Zähler laden
1917  2D                LOOP3:  DEC     L               ;Zähler dekrementieren
1918  20 FD                     JR      NZ,LOOP3        ; bis Zählerstand = 00
191A  25                        DEC     H               ;Zähler dekrementieren
191B  20 F8                     JR      NZ,LOOP4        ; bis Zählerstand = 00
191D  E1                        POP     HL              ;HL zurückholen
191E  C9                        RET                     ;Ende des Unterprogramms

                        ;*****************************************************
```

PER14 S91/ED1 :3 (AN7=2)
S 91 L 75

```
                        ;*****************************************************
                        ;*                                                   *
                        ;*    Einfaches Programm für 8255 in Betriebsart 1   *
                        ;*                                                   *
                        ;*****************************************************

                        ;Definitionen:

                        PORTA   EQU     0C0H            ;Port A 8255 Peripherie
                        PORTC   EQU     0C2H            ;Port C 8255 Peripherie
                        STW     EQU     0C3H            ;Steuerwortregister 8255
                        STW1    EQU     10110111B  B7H  ;Steuerwort Betriebsart1
                        STW2    EQU     01111001B  79H  ;Inerruptfreigabe für A

                        HEX7    EQU     0689H           ;Codewandlerprogramm
                        SCAN    EQU     0624H           ;Anzeigeprogramm
                        USER EQU 1FEEH
                                ORG     1800H

                        ;*****************************************************

                        ;Hauptprogramm

                                                INTER
1800  21 50 19          START:  LD      HL,1950H        ;Anfangsadresse
1803  22 EE 1F                  LD      (1FEEH),HL      ;  der Interr.-Routine
                                                USER
1806  CD 50 18                  CALL    INIT            ;8255 initialisieren

1809  FB                        EI                      ;Interrupt freigeben

180A  3E 0F             LOOP1:  LD      A,0FH           ;Anfangsstand Zähler
180C  F5                LOOP2:  PUSH    AF              ;Akku retten
180D  CD 89 06                  CALL    HEX7            ;Codewandlerprogramm
1810  32 05 19                  LD      (1905H),A       ;Byte nach Anz.-Puffer
1813  F1                        POP     AF              ;Akku zurückholen

1814  CD 80 18                  CALL    ANZEI           ;Anzeigeprogramm
1817  CD 70 18                  CALL    ZEIT            ;Zeitverzögerung

181A  3D                        DEC     A               ;Zähler abwärtszählen
181B  20 EF                     JR      NZ,(LOOP2)      ;  bis (Zähler) = 00
181D  C3 0A 18                  JP      (LOOP1)

1820                    ;*****************************************************

                                ORG     1850H

1850  3E B7            INIT:   LD      A,STW1          ;Steuerwort in den Akku
1852  D3 C3                    OUT     (STW),A         ;und an das Steuerwort-
                                                      ;register weitergeben
1854  3E 79                    LD      A,STW2          ;Steuerwort für INTE FF
1856  D3 C3                    OUT     (STW),A         ;an das Steuerwortreg.

1858  C9                        RET                     ;Ende des Unterprogramms

                        ;*****************************************************
```

```
                        ;*****************************************************

                        ORG     1870H

1870  E5        ZEIT:   PUSH    HL              ;HL auf den Stack retten

1871  26 2F             LD      H,2FH           ;Zähler laden
1873  2E AF       LOOP4: LD      L,0AFH          ;Zähler laden
1875  CD 80 18          CALL    ANZEI           ;Anzeigeunterprogramm
1878  2D         LOOP3: DEC     L               ;Zähler abwärtszählen
1879  20 FD              JR      NZ,(LOOP3)      ;  bis (Zähler) = 00
187B  25                DEC     H               ;Zähler abwärtszählen
187C  20 F5             JR      NZ,(LOOP4)      ;  bis (Zähler) = 00

187E  E1                POP     HL              ;HL vom Stack zurück
187F  C9                RET                     ;Ende des Unterprogramms

                        ;*****************************************************

                        ORG     1880H

1880  F5         ANZEI: PUSH    AF              ;alle Registerinhalte
1881  C5                PUSH    BC              ;  retten
1882  D5                PUSH    DE
1883  E5                PUSH    HL

1884  DD 21 00 19       LD      IX,1900H        ;Anfangsadr.des Puffers
1888  CD 24 06          CALL    SCAN            ;Unterprogramm Anzeige

188B  E1                POP     HL              ;alle Registerinhalte
188C  D1                POP     DE              ;  vom Stack zurück
188D  C1                POP     BC
188E  F1                POP     AF

188F  C9                RET                     ;Ende des Unterprogramms

                        ;*****************************************************

                        ORG     1950H

1950  F3         INTER: DI      NOP             ;erneuten Interrupt
                                                ;  sperren.
1951  DB C0              IN      A,(PORTA)       ;Byte von Port A wird
1953  CD 89 06          CALL    HEX7            ;  7-Segm.-Code: In den
1956  32 00 19          LD      (1900H),A       ;  den Anzeigepuffer.

1959  FB                EI                      ;Interrupt freigeben

195A  ED 4D             RETI    NOP             ;Ende der Interrupt-
                                RET             ;Routine

                        ;*****************************************************
```

```
                ;*****************************************************
                ;*                                                   *
                ;* Ausgabe eines Speicherbereichs über den Port A    *
                ;*        des 8255 in der Betriebsart 1.             *
                ;*  Mit jedem Interrupt wird der Inhalt einer        *
                ;*      Speicherzelle an den Port C geliefert.       *
                ;*                                                   *
                ;*****************************************************

                ;Definitionen:

                PORTA   EQU     0C0H            ;Port A 8255 Peripherie
                PORTC   EQU     0C2H            ;Port C 8255 Peripherie

                STWR    EQU     0C3H            ;Steuerwortregister 8255
                STW     EQU     10100101B       ;Steuerwort für 8255
                STW1    EQU     01111101B       ;   "   Freigabe INTE A

                BIT     EQU     01111001B       ;Bereitsignal an Port C
                        ORG     1800H

                ;*****************************************************

                ;Hauptprogramm

1800  3E A5       START:  LD      A,STW           ;Steuerwort in das
1802  D3 C3               OUT     (STWR),A        ;  Steuerwortregister

1804  21 00 19           LD      HL,1900H        ;Anfangsadresse der
1807  22 EE 1F           LD      (1FEEH),HL      ;  Interrupt-Routine

180A  06 10              LD      B,10H           ;Zähler laden
180C  21 50 19           LD      HL,1950H        ;Anfangsadresse Puffer

180F  3E 79              LD      A,BIT           ;Bereitsignal an Port C
1811  D3 C3               OUT     (STWR),A        ;  ausgeben

1813  3E 7D              LD      A,STW1          ;INTE A freigeben
1815  D3 C3               OUT     (STWR),A        ;

1817  FB                 EI                      ;Interrupt freigeben
1818  00         LOOP:   NOP                     ;Auf ein Interrupt-Si-
1819  C3 18 18           JP      LOOP            ;gnal warten

                ;*****************************************************
```

```
                        ;*****************************************************

                                ORG     1900H

1900  F3                INTERR: DI              ;Interrupt sperren

1901  7E                        LD      A,(HL)  ;Daten aus dem Puffer
1902  D3 C0                     OUT     (PORTA),A ;  an Port A

1904  CD 0F 19                  CALL    ZEIT    ;Ca. 1 Sekunde warten

1907  23                        INC     HL      ;Nächste Stelle Puffer
1908  05                        DEC     B       ;Zähler dekrementieren
1909  28 03                     JR      Z,(HALT) ;Programmende bei B = 00

190B  FB                        EI              ;Interrupt freigeben
190C  ED 4D                     RETI            ;Ende Interr.-Routine

190E  76                HALT:   HALT            ;Programmende

                        ;*****************************************************

190F  16 FF             ZEIT:   LD      D,0FFH  ;Zähler 1 laden

1911  1E AF             LOOP2:  LD      E,0AFH  ;Zähler 2 laden
1913  1D                LOOP1:  DEC     E       ;Zähler 2 dekrementieren
1914  20 FD                     JR      NZ,(LOOP1) ;  bis (Zähler 2) = 00
1916  15                        DEC     D       ;Zähler 1 dekrementieren
1917  20 F8                     JR      NZ,(LOOP2) ;  bis (Zähler 2) = 00
1919  C9                        RET             ;Ende des Unterprogramms

                        ;*****************************************************
```