SORTIERPROGRAMME : BUBBLE SORT

Eine einfache Methode um eine Liste von Daten der Größe nach zu sortieren, besteht darin, immer zwei Werte der Liste miteinander zu vergleichen und zu vertauschen, wenn sie nicht in der richtigen Reihenfolge sind. Zunächst werden die beiden letzten Werte der Liste verglichen und vertauscht, wenn die Reihenfolge nicht stimmt. Der vorletzte Wert wird dann mit dem folgenden verglichen und - wenn nötig - getauscht. Solange nicht getauscht zu werden braucht, wird in Richtung Listenanfang sortiert. Der Rechner bewegt sich wie eine vom Boden eines Gewässers aufsteigende Blase (bubble = engl. Blase) durch die Daten hindurch. Jedesmal wenn ein Tausch erforderlich war, wird allerdings die Sortierrichtung umgekehrt. Jetzt wird der Teil der Liste, der schon sortiert wurde, solange umsortiert, bis der größere Wert seine richtige Position gefunden hat. Erst dann wird der Sortiervorgang in der entgegengesetzten Richtung wieder aufgenommen, und zwar an der Stelle, an der der größere Wert gefunden wurde.
Dieser "bottom - to - top bzw. top - to - bottom" Mechanismus wird solange durchgeführt, bis alle Werte in der richtigen Reihenfolge vorliegen. Es leuchtet ein, daß die reine Rechenzeit nicht unbedingt von der Länge der Liste abhängig ist, sondern im wesentlichen von der Anzahl der durchzuführenden Tauschoperationen. Im allgemeinen ist "Bubble - Sort" ein langsames Sortierverfahren. Es kann aber auch sehr schnell sein, nämlich dann, wenn die Liste der Daten schon vorsortiert ist.

Ein einfaches Beispiel mit 5 Zahlenwerten soll das Prinzip von BUBBLE SORT zeigen. Nach 7 Sortiervorgängen ist die Liste sortiert.

1.	2.	3.	4.	5.	6.	7.
3	3	3	3	3	3+	0
5	5	5+	0	0	0	3
0	0	0	5+	4	4	4
6+	4	4	4	5	5	5
4	6	6	6	6	6	6

\+ = Tausch erforderlich.

Das umseitige Listing ist ein BUBBLE SORT Übungsprogramm.
Bei den bedingten Sprungbefehlen wurde nur relative Adressierung verwendet. Das Programm kann also in einem beliebigen Speicherbereich stehen. Es besteht allerdings die Einschränkung, daß die zu sortierenden Daten im Bereich 0B00 - 0BFF stehen sollen, wenn das Programm fest in einem EPROM abgespeichert ist.
Die Daten werden in aufsteigender Folge sortiert, d.h. in 0B00 steht der kleinste, in 0BFF der größte Wert, wenn 256 Werte vorliegen. Vor Programmbeginn wird mit der Speicherstelle 08FF bestimmt, aus welchem Zahlenbereich die Daten stammen. Bei 00 sind die Werte 7-Bit-Zahlen mit Vorzeichen, bei 02 sind 8-Bit-Zahlen aus dem logischen Zahlenbereich zu sortieren. Mit 00 bzw 02 wird das COMPARE-FLAG im UNTEREN PSW initialisiert.

BEISPIELPROGRAMM: BUBBLE SORT I

Das Programm sortiert 8 Bit-Werte in aufsteigender Reihenfolge. Die Liste darf max. 256 Werte lang sein und steht im festen Speicherbereich 0B00 - 0BFF. Das COMPARE FLAG muß 1 gesetzt werden, wenn die Daten aus dem logischen Zahlenbereich kommen, bei COM = 0 werden sie als vorzeichenbehaftete Zweierkomplemente interpretiert. Der für das COMPARE FLAG gewünschte Wert ist vor Programmstart in die Speicherstelle 08FF zu schreiben.

Register R2 arbeitet als Indexregister und wird mit der Listenlänge minus 1 geladen. Bei der maximalen Listenlänge von 256 Werten ist dies der Wert HFF!
Register R3 zählt die Programmdurchläufe bzw die Tauschvorgänge. Wenn R3 auf Null herabgezählt ist - also kein Tausch mehr erforderlich war - wird das Sortierprogramm beendet. Der letzte Befehl des Programms kann ein HALT - oder ein RETURN - Befehl sein. Bei einem RETURN (Code 17) ist BUBBLE SORT als Unterprogramm aufzurufen.

Die Anfangsadresse 0B00 der Liste erhält die symbolische Bezeichnung 'Liste', die Adresse 0AFF ist dann 'Liste-1'.

```
LABEL  ADR   OPCODE   ASSEMBLER              KOMMENTAR

       08FF  00 oder 02
Start  0900  0C 08 FF LODA,R0 '08FF'         COM - Flag setzen
       0903  93       LPSL
       0904  07 FF    LODI,R3 FF            Listenlänge -1 nach R3
Sort   0906  03       LODZ,R3               (R3) über R0 nach R2
       0907  C2       STRZ,R2               umspeichern
Loop   0908  0E 6B 00 LODA,R0 'Liste',R2    1. Wert laden und mit
       090B  EE 6A FF COMA,R0 'Liste-1',R2  2. Wert vergleichen
       090E  9A 11    BCFR,LT 'Ende'        Sprung wenn Reihenfolge
       0910  C1       STRZ,R1               in Ordnung, sonst Werte
       0911  0E 6A FF LODA,R0 'Liste-1',R2  im Speicher vertauschen
       0914  CE 6B 00 STRA,R0 'Liste',R2
       0917  01       LODZ,R1
       0918  CE 6A FF STRA,R0 'Liste-1',R2
       091B  DA 00    BIRR,R2               Indexregister erhöhen und
       091D  E6 00    COMI,R2 00            prüfen ob Listenende
       091F  98 67    BCFR,EQ 'Loop'        erreicht, nein, weiter
Ende   0921  FB 63    BDRR,R3 'Sort'        sonst in Richtung Listen-
       0923  40       HALT                  anfang sort. bis R3=0
```

Die im Listing unterstrichenen Opcodes müssen geändert werden, wenn das Programm auf eine andere Anfangsadresse der Liste eingestellt werden soll. Die Listenlänge kann dabei durch den Wert in 0905 und 091E bestimmt werden. Nehmen wir einmal an, es sind lediglich 6 Werte (z.B. Lottozahlen) zu sortieren! Die Adresse 0905 erhält dann den Wert 05 (Listenlänge - 1), 091E wird mit dem Wert 06, also der Listenlänge, geladen.

Beispielprogramm: BUBBLE SORT II

Das folgende Programm verwendet denselben Sortieralgorithmus wie das umseitige Programm. Es kann max. 256 8-Bit Werte sortieren. Das Programm ist aber nicht auf eine feste Anfangsadresse der Daten und eine feste Listenlänge eingeschränkt. Anfangsadresse und Listenlänge werden vom Programm automatisch aus festgelegten RAM - Adressen erkannt und geladen. Das Programm selbst kann in ein ROM geschrieben und universell als UP verwendet werden. Die Start-adresse im EPROM ist 1500. Die 4 RAM-Adressen sind:

Adresse 0830 : enthält das High-Byte der Anfangsadresse der Liste
Adresse 0831 : enthält das Low-Byte der Anfangsadresse der Liste
Adresse 0832 : enthält den Wert der Listenlänge minus 1
Adresse 0833 : FLAG - 00 bei arithmetischen, 02 bei logischen Werten

Beispiel: 20 Werte aus dem arithmetischen Zahlenbereich, die im Speicher ab Adresse 0950 abgespeichert sind, sollen in aufsteigender Folge sortiert werden. Die vier Adressen erhalten folgende Werte vor dem Programmstart:

0830 = 09 High-Byte der Adresse
0831 = 50 Low-Byte der Adresse
0832 = 13 Hex-Wert für 19 (Listenlänge - 1)
0833 = 00 Flag für arithmetischen Vergleich

Das Programm arbeitet mit indirekter indizierter Adressierung beim Aufspüren der Daten in der Liste. Die Adressen 0830 und 0831 dienen also als Zeiger (engl. pointer) auf die eigentliche Datenadresse im Speicher (hier 0950).

```
LABEL ADR  OPCODE   ASSEMBLER          KOMMENTAR

Start 1500 0C 08 33 LODA,R0 '0833'     log. oder arithm. Vergleich
      1503 93       LPSL               vorschreiben
      1504 0F 08 32 LODA,R3 '0832'     Durchlaufzähler R3 laden
Sort  1507 FB 00    BDRR,R3 '1509'     und dekrementieren
      1509 03       LODZ,R3            R2 über R0 als Indexregister
      150A C2       STRZ,R2            mit (R3) laden
Loop  150B 0E E8 30 LODA,R0 *'0830'R2  1. Wert nach R0 laden
      150E EE A8 30 COMA,R0 *'0830'R2+ mit 2.Wert vergleichen
      1511 99 10    BCFR,GT 'Ende'     Sprung wenn Reihenfolge i.O.
      1513 C1       STRZ,R1            sonst Werte tauschen
      1514 0E E8 30 LODA,R0 *'0830'R2
      1517 CE C8 30 STRA,R0 *'0830'R2-
      151A 01       LODZ,R1
      151B CE A8 30 STRA,R0 *'0830'R2+
      151E EE 08 32 COMA,R2 '0832'     (R2) mit Listenlänge vergl.
      1521 98 62    BCFR,EQ 'Loop'     Listenende? Nein, Sprung
Ende  1523 5B 62    BRNR,R3 'Sort'     Sprung wenn Progr. nicht
      1525 17       RETC,UN            fertig, sonst zurück ins
                                       Hauptprogramm
```

DIE PROGRAMMIERUNG DER PORTS: ERWEITERTE SCHREIB- UND LESEBEFEHLE

Ein **Port** ist ein Zugang zum Datenbus, der es ermöglicht, dem Datenbus Informationen zu entnehmen und festzuhalten oder auf ihn zu schreiben. Der Portbaustein mit der **Adresse 09** ist ein **AUSGANGSPORT**. Die Daten werden nach einem Ausgabebefehl für diesen Port solange festgehalten, bis sie überschrieben werden oder ein **RESET** ausgelöst wurde. Der binäre Wert der Daten wird über die 8 Portleds optisch angezeigt. Der Portstecker gestattet den Anschluß von **INTERFACESCHALTUNGEN** für die Ansteuerung von Motoren, Elektromagneten, Lautsprechern und kompletten Funktionsmodellen. In den folgenden Kapiteln wird z.B. eine einfache Ampelsteuerung und die Steuerung von ft-Modellen aus dem **computing** Baukasten beschrieben.

Der Portbaustein mit der **Adresse 08** ist ein **Universalport**. Man kann in ihn schreiben und ihn lesen. Wir benutzen ihn überwiegend als **Eingabeport**. Über den Portstecker können z.B. Taster an den Port angeschlossen werden. Diese können vom Programm auf 0 oder 1 abgefragt werden und für Steuerungszwecke benutzt werden. Port 08 unterscheidet sich bei der Ausgabe von Port 09. Die Daten erscheinen nur für die Dauer des Ausgabebefehls (einige Mikrosekunden) an den Portausgängen. Sie werden also nicht zwischengespeichert. Eine optische Anzeige entfällt, sie wäre sowieso nicht wahrnehmbar.

Ein weiterer Universalport befindet sich auf der Tastaturplatine. Er wird 7 - fach benutzt. Über die **Portadresse 17** kann die **Hex-Tastatur gelesen** werden. Die **Portadressen 10 - 15** sind für die sechs Ziffernanzeigen der **Anzeigeeinheit** reserviert. Jedem Digit ist eine Portadresse zugeordnet:

Digit 0 (ganz rechts):	10
1	11
2	12
3	13
4	14
5 (ganz links):	15

Die Schaltungstechnik der Portschaltungen ist im Begleitbuch in den Kapiteln 10 und 12 ausführlich beschrieben.

Für den Anschluß einer **A/D-Wandlerplatine** ist die **Portadresse 0A** vorgesehen. Die Decodierung dieser Adresse wird auf der Portplatine vorgenommen, der eigentliche Portbaustein befindet sich auf der A/D-Wandlerplatine. Da der A/D -Wandler vom Programm gelesen wird, handelt es sich hier auch um einen Eingabeport.

Die oben beschriebenen Ports bzw ihre Adressen sind für die gesamte Hardware des Systems ausreichend. Für Erweiterungen stellt die Portplatine die ungenutzten Portadressen zur Verfügung. Es sind die Adressen 00, 01, 02, 03, 04, 05, 06, 07, 0B, 0C, 0D, 0E und 0F.

In Mikroprozessorprogrammen finden sich sehr häufig Strukturen, die aus einer Portabfrage (READ), einem oder mehreren Vergleichsbefehlen (COMPARE) und daraus resultierenden Programmverzweigungen (BRANCH) bestehen. Diese Grundstrukturen sollen an kleinen Beispielprogrammen erläutert werden.

BEISPIEL 1: Portadressierung - Ausgeben über Port 09

Die Zahl H'55' soll nach Register R0 geladen und dann zum Port 09 transportiert werden. Dann wird der Prozessor angehalten.

Der Wert 55 wird zunächst mit einem LODI-Befehl in das gewählte Register geladen. Für die **Ausgabe** des Registerinhaltes verwenden wir im Grundsystem mit Speicher und Port die **erweiterten Schreibbefehle WRTE,r**. Die Assemblerschreibweise ist die Abkürzung der englischen Befehlsbeschreibung "**write extended**". Es handelt sich um einen 2-byte-Befehl. Das erste Byte enthält den eigentlichen **Ausgabebefehl** mit der Codierung des Registers (Bit 0 und 1), dessen Inhalt ausgegeben werden soll. Das zweite Byte enthält die **Adresse des Ports**, für den die Daten bestimmt sind. Aus den 256 Möglichkeiten haben wir durch unsere Hardware die Adresse 09 ausgewählt. Jetzt läßt sich mit Hilfe der Befehlsliste (S.61) der vollständige Befehl codieren.

```
WRTE,R0 H'09'   D4 09    % Schreibe den Inhalt von R0 in den
                           Port mit der Adresse 09.
```

Die Ausgabebefehle für R1, R2 und R3 folgen der schon bekannten Systematik:

```
WRTE,R1 H'09'   D5 09
WRTE,R2 H'09'   D6 09
WRTE,R3 H'09'   D7 09
```

Der Befehl **HALT** (40 hex) schließt das Programm ab. Zur Erinnerung: Wenn der Prozessor in den HALT-Zustand läuft, erlischt die LED RUN/WAIT auf der CPU-Platine und sämtliche LEDs der Daten- und Adresseneingabe leuchten auf! Die einzige Möglichkeit um wieder auf RUN umzuschalten, ist das Betätigen des RESET-Tasters.

Wir **laden** das folgende Programm in den Speicher und **starten** es gemäß der **Bedienungsanleitung** für das Grundsystem. Bitte nicht vergessen, den Schalter auf der Speicherplatine in die Stellung RAM zu bringen! Die Anfangsadresse des Programms ist dann die Adresse 00 00 (hex). Die Hex-Tastatur können wir jetzt natürlich nicht verwenden. Als Taktfrequenz wählen wir 1Hz.

ADR	DATEN	BEFEHL	KOMMENTAR
0000	04 55	LODI,R0 H'55'	Lade R0 mit 55
0002	D4 09	WRTE,R0 H'09'	Schreibe (R0) in den Port 09
0003	40	HALT	Beende das Programm

Wenn keine Bedienungsfehler gemacht wurden, erscheinen die Daten 55 an den 8 Portleds (0101 0101).
Das kleine Programm läßt sich abwandeln, indem man andere Werte nach R0 lädt bzw die anderen CPU-Register verwendet.

BEISPIEL 2: Portadressierung - Lesen von Port 08 und Ausgeben über Port 09. Programmieren einer Endlosschleife. An die Datenleitung D0 des Portsteckers für Port 08 soll ein Eingabetaster (z.B. Fischertechnik) angeschlossen werden. Der Taster wird über Port 08 fortwährend gelesen und sein Wert über Port 09 angezeigt.

Der Taster (Schließer) wird mit einem Anschluß mit der **GND-Leitung** des Computers verbunden, der zweite Anschluß wird zum **Portstecker** für **Port 08** geführt und dort eingepreßt. Die Belegung des Steckers ist im Begleitbuch auf S.187 beschrieben. Wenn man für den Anschluß des Tasters nicht einen ganzen Portstecker opfern will, kann man auch in die Lötpunkte auf den Leiterbahnen vom Portbaustein (IC 74LS245) zum Portstecker Lötstifte einlöten. Die Verbindung wird dann mit passenden Steckschuhen hergestellt. Die Lötpunkte waren ursprünglich für die acht 1k Ohm - Ableitwiderstände vorgesehen, die man stehend montieren sollte. Diese Widerstände sollte man jedoch besser auf einer Interfaceschaltung anbringen. In unserem Fall stören sie sogar, weil sie eine feste Verbindung eines Porteingangs mit GND herstellen. Beim Lesen würden die Porteingänge also immer den Wert 0 liefern und einen geschlossenen Taster simulieren. Die Datenleitung für **Bit 0** führt zum **Pin 9 am 74LS245, Bit 7** ist am **Pin 2** abgreifbar.

Für das **Lesen (READ)** eines Ports verwenden wir die **erweiterten Lesebefehle (READ EXTENTED)**. Es sind 2-Byte Befehle. Das erste Byte enthält den eigentlichen Lesebefehl und die binäre Codierung des Registers, in das der eingelesene Binärwert transportiert werden soll. Das 2.Byte enthält die Adresse des Ports aus dem gelesen werden soll. Da für die Portabfrage alle Register zugelassen sind, erhalten wir vier Lesebefehle. Den Assemblercode finden wir auf S.347 im Begleitbuch.

```
REDE,R0 H'08'      54 08    * Lies den Port mit der Adresse 08
REDE,R1 H'08'      55 08    * in das im Befehl genannte Register
REDE,R2 H'08'      56 08    *
REDE,R3 H'08'      57 08    *
```

Unmittelbar nach dem Lesen des Ports 08 wird der Inhalt des verwendeten Registers mit einem WRTE-Befehl (s.o.) in den Port 09 geschrieben. Unbeschaltete Porteingänge erzeugen eine logische **Eins**. Mit dem an Bit 0 angeschlossenen Taster werden logische **Nullsignale** erzeugt, wenn der Taster betätigt wird. Die LED D0 an Port 09 erlischt, während die übrigen leuchten. Mit einem **unbedingten Sprungbefehl** können wir eine Endlosabfrage programmieren. Nach dem Ausgabebefehl verzweigt das Programm an den Anfang und liest Port 08 neu ein. Der verwendete **BCTA-Befehl** (BRANCH ON CONDITION TRUE, ABSOLUTE) ist 3 Byte lang. Das erste Byte enthält die Codierung von "Springe unbedingt", das zweite und dritte Byte geben die vollständige Adresse des Sprungziels an. Im Beispiel lautet der Hexcode **1F 00 00**. Die große Gruppe der **Sprungbefehle** wird noch ausführlich behandelt.

Listing des 2. Beispiels:

```
ADR    DATEN     BEFEHL                KOMMENTAR

0000   54 08     REDE,R0 H'08'         Port 08 nach R0 einlesen
0002   D4 09     WRTE,R0 H'09'         (R0) über Port 09 anzeigen
0004   1F 00 00  BCTA,UN '0000'        unbedingt zum Anfang zurück
```

BEISPIEL 3: ABFRAGE DES TASTATURPORTS UND ANZEIGE DER BINÄREN TASTENWERTE ÜBER PORT 09.

Das Grundsystem wird um die Hex-Tastatur erweitert. Auf der Speicherplatine wird das EPROM mit dem Betriebssystem eingeschaltet. Die Anfangsadresse des Beispielprogramms muß jetzt 0900 sein. Die Bedienung der Tastatur ist im Begleitbuch ab S.234 ausführlich beschrieben.

Das Beispielprogramm unterscheidet sich von Beispiel 2 lediglich durch die Portadresse (17 für 08). Bei jedem Tastendruck - die Taste muß festgehalten werden - wird an den 8 Portleds der Binärwert der entsprechenden Taste angezeigt. Die Taste RST kann nicht abgefragt werden. Die Tastenwerte sollte man sich in einer kleinen Tabelle notieren, weil sie in vielen Übungsprogrammen noch benötigt werden.

```
ADR    DATEN     BEFEHL                KOMMENTAR

0900   54 17     REDE,R0 H'17'         Tastenport nach R0 lesen
0902   D4 09     WRTE,R0 H'09'         (R0) ausgeben
0904   1F 09 00  BCTA,UN '0900'        weiter abfragen
```

TABELLE DER TASTENWERTE (HEX)

Taste	Wert	*	Taste	Wert	*	Taste	Wert	*	Taste	Wert	*
0		*	6		*	C		*	GOTO		*
1		*	7		*	D		*	F		*
2		*	8		*	E		*	MON		*
3		*	9		*	F		*	PC		*
4		*	A		*	CMD		*	NXT		*
5		*	B		*	RUN		*			*

Aus der Tabelle erkennen wir, daß bei jedem Tastenwert das höchstwertige Bit 7 gesetzt ist. Bit 7 hat eine Art Kontrollfunktion, es wird dann "Eins", wenn überhaupt eine Taste gedrückt wurde. Die restlichen 7 Bits dienen zur binären Codierung der einzelnen Tasten. Man erkennt leicht, daß bei nur 23 Tasten nicht alle möglichen Bitkombinationen ausgenutzt werden. Bei der Programmierung von Tastaturabfragen ist es üblich, zunächst nur das Kontrollbit auf logisch 1 zu prüfen und erst dann die gedrückte Taste zu identifizieren. Vergleichen sie dazu auch Kapitel 12!

BEISPIEL 4: ANSTEUERUNG DES DISPLAYS

Das Wort "Ferien" soll auf dem sechsstelligen Display angezeigt werden.

Die Anzeigeeinheit besteht aus 6 Siebensegmentanzeigen. Jede einzelne Anzeige ist aus 8 (!) LEDs aufgebaut. Sieben Segmente ermöglichen dabei die Darstellung der Ziffern 0 - 9 und einiger Großbuchstaben des Alphabets (z.B. A, E, F). Das achte Segment ist der Dezimalpunkt (dp). Mit etwas Phantasie kann man sogar das gesamte Alphabet darstellen, man muß lediglich einige Zeichen doppelt belegen (1 = i; 5 = S; 6 = G usw) und Groß- und Kleinschreibung mischen. Insgesamt lassen sich 45 sinnvolle Zeichen erzeugen. Der Hexcode dieser Zeichen reicht von 00 - 2C. Der Zeichensatz ist im Begleitbuch auf S.215 dargestellt.

Zu jedem Zeichen gehört ein Siebensegmentcode. Das ist der Binärwert, der über die Portadresse zur Ansteuerung einer Anzeige ausgegeben werden muß. Die Werte werden leicht verständlich, wenn man sich vorstellt, daß jeder Datenleitung D0 - D7 ein Segment der Anzeige zugeordnet ist:

D0 = Segment a
D1 = Segment b
D2 = Segment c
D3 = Segment d
D4 = Segment e
D5 = Segment f
D6 = Segment g
D7 = Segment dp

Die Anordnung der Segmente a - dp ist genormt, nicht aber die Zuordnung unserer Datenleitungen. Ein Segment wird durch eine "1" auf der angeschlossenen Datenleitung zum Aufleuchten gebracht, eine "0" läßt das betreffende Segment dunkel.

Um das Zeichen "F" aus unserem Beispiel "FERIEN" zu erzeugen, sind die Segmente a, e, f und g zu aktivieren. Dazu müssen die Datenleitungen D0, D4, D5 und D6 logisch 1 sein. Es ergibt sich der Binärwert 0111 0001 (71 hex) als Siebensegmentcode für das Zeichen "F". Mit Hilfe des kompletten Zeichensatzes, der auf der nächsten Seite aufgelistet ist, ermitteln wir die Zeichencodes für "E", "R", "I" und "N".

E (Segmente a, f, e, g und d) = 79
R (Segmente g und e) = 50
I (Segmente b und c) = 06
N (Segmente c, e und g) = 54

Durch die Schaltung der Anzeigeeinheit sind dem Programmierer bestimmte Programmschritte vorgegeben, die er einhalten muß, wenn eine Anzeigeroutine funktionieren soll.

Hier jetzt einige Punkte, die bei der Programmierung der Anzeige zu beachten sind:

1. Die Daten für ein Digit sind nur für die Dauer von zwei CLOCK-Pulsen während des Schreibbefehls gültig. Sie werden nicht in einem Auffangregister - ähnlich Port 09 - zwischengespeichert.
2. Die 6 Digits müssen nacheinander angesteuert werden.
3. Die Anzeigen müssen nacheinander so schnell zum Aufleuchten gebracht werden, daß das Auge den Wechsel nicht mehr erkennt und den Eindruck einer Daueranzeige hat. Diese Ansteuerungstechnik nennt man Multiplex - Betrieb.
4. Anzeigeroutinen müssen daher in längeren Programmen sehr sorgfältig in Schleifen eingebaut werden.
5. Anzeigeroutinen benötigen Zeit, in der der Rechner für andere Tätigkeiten blockiert ist.
6. Es ist meistens günstig, wenn man Anzeigeroutinen als Unterprogramme schreibt und dabei das Betriebssystem im EPROM verwendet.

Das kleine Übungsprogramm beachtet besonders die Punkte 1 - 3. Das Arbeitsregister R0 wird mit Hilfe der LODI - Befehle mit den Siebensegmentcodes der gewünschten Zeichen geladen. Mit einem erweiterten Schreibbefehl WRTE wird der Zeichencode über die Ports 10 - 15 an die einzelnen Digits ausgegeben. Nachdem alle Digits einmal angesteuert wurden (für ca 2 Mikrosekunden!), verzweigt das Programm mit einem unbedingten Sprungbefehl wieder zum Anfang. Nur durch diese Endlosschleife läßt sich eine Daueranzeige des Wortes "Ferien" erreichen.

PROGRAMMLISTING:

```
ADR     DATEN       BEFEHL              KOMMENTAR

0900    04 71       LODI,R0 H'71'       Zeichencode für "F" laden
0902    D4 15       WRTE,R0 Port 15     an Digit 5 (links) ausgeben
0904    04 79       LODI,R0 H'79'       Zeichencode für "E" laden
0906    D4 14       WRTE,R0 Port 14     an Digit 4 und 1 ausgeben
0908    D4 11       WRTE,R0 Port 11
090A    04 50       LODI,R0 H'50'       Zeichencode für "r" laden
090C    D4 13       WRTE,R0 Port 13     an Digit 3 ausgeben
090E    04 06       LODI,R0 H'06'       Zeichencode für "I" laden
0910    D4 12       WRTE,R0 Port 12     an Digit 2 ausgeben
0912    04 54       LODI,R0 H'54'       Zeichencode für "n" laden
0914    D4 10       WRTE,R0 Port 10     an Digit 0 ausgeben
0916    1F 09 00    BCTA,UN '0900'      Programm wiederholen
```

Wir starten das Programm mit RUN und erhalten die gewünschte Anzeige. Mit diesem kleinen Anzeigeprogramm kann man jetzt etwas spielen. Man könnte z.B. seinen Vornamen anzeigen, die Uhrzeit oder das Datum ausgeben ...