Telefonservice
08 31- 62 11
jeden Mittwochabend
bis 20.00 Uhr

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang
Telefon: 08 31 / 6 93 00

Geschäftszeiten: GES GmbH + Verkauf
Mo. - Do. 8.00 - 12.00 Uhr, 13.00 - 17.00 Uhr
Freitag 8.00 - 12.00 Uhr
Telefonservice

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr

Handbuch

Einsteigerpaket

Graf Elektronik Systeme GmbH

Inhaltsverzeichnis

Literatur zum Einsteigerpaket

1. Handbuch zum TOOL EPROM

Ihr Einsteigerpaket ist nicht nur dazu geeignet, Programme zu entwickeln und auf der HEXIO2 irgendwelche Leuchtdioden zum Leuchten zu bringen, sondern sie können auch zahlreiche IO-Baugruppen an ihr Einsteigerpaket anschließen. Die Software für diese Baugruppen ist auf dem EPROM EINDHEX V1.2 schon enthalten.

Folgende Baugruppen werden unterstützt:

- AD 8*16: Analog Digital Wandler, 8 Bit Auflösung, 16 Kanäle
- AD 10*1: Analog Digital Wandler, 10 Bit Auflösung, 1 Kanal
- Ampel: Demo-Baugruppe Ampelsteuerung; wird auf die IOE aufgesteckt
- CENT: Centronics-Druckerschnittstelle, z.B. Ausdruck eines Hexdumps
- D/A: Digital Analog Wandler, 8 Bit Auflösung, 2 Kanäle
- REL: Relais-Baugruppe; steuert acht Relais
- SER: Serielle Schnittstelle, z.B. für seriellen Drucker
- SOUND: Sound-Baugruppe; drei programmierbare Tongeneratoren und ein Rauschgenerator
- SPRACHE: Sprache-Baugruppe; mit der Baugruppe kann Sprache erzeugt werden; 64 Phoneme, Lautstärke, Phonemdauer usw. programmierbar; spezielle Filter, um der Sprache eine gewisse Persönlichkeit zu verleihen.

Die Software für die obengenannten Baugruppen ist im EPROM EINDHEX V1.2 bereits enthalten. Es gibt selbstverständlich auch eine Beschreibung (komplett mit Struktogrammen und dokumentiertem Quellisting der Programme) zu dieser Software, die Sie bei Graf Elektronik Systeme unter der Bestellnr. 11185 beziehen können. Auskünfte und Preise über die obengenannten Baugruppen erhalten Sie ebenfalls bei Graf Elektronik Systeme, Magnusstr. 13, 8960 Kempten (Tel. 0831/6211).

2. "Mit HEXMON Programme entwickeln"

Ein weiteres nützliches Hilfsmittel, um mit dem Einsteigerpaket Programme zu entwickeln, ist das Buch von Rolf Dieter Klein "Mit HEXMON Programme entwickeln". Dieses Buch beschreibt den im EPROM EINDHEX2 V1.2 befindlichen Monitor. Hierin finden Sie eine ausführliche Beschreibung der Monitorroutinen, Programmbeispiele, Hardwarebeschreibung und zusätzlich noch das dokumentierte Listing des Monitors selbst. Dieses Buch ist für den Einsteiger der etwas tiefer einsteigen will wirklich unentbehrlich.

Dieses Buch können Sie ebenfalls über Graf Elektronik Systeme unter der Bestellnummer 10286 beziehen.

Zusammenfassung

Art. Nr.	Bezeichnung	Preis (freibleibend inc. MWST)
11185	ETOOL Handbuch	20,-- DM
10286	Buch: Mit HEXMON Programme entwickeln	20,-- DM

Herzlichen Glückwunsch

Hier liegt es endlich vor Ihnen, Ihr Einsteigerpaket. Da Sie wahrscheinlich darauf brennen, mit Ihrem System zu arbeiten, fassen wir uns so kurz wie möglich. Bevor Sie aber wild drauf los programmieren, sollten Sie doch die nächsten Seiten lesen.

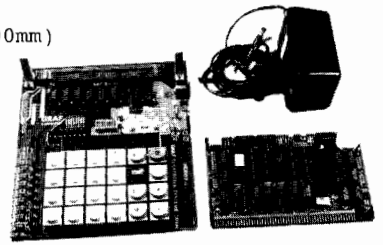
Wir packen aus

Verfahren Sie hierbei bitte nach dem Motto:

"Vertrauen ist gut, Kontrolle ist besser!"

Prüfen Sie, ob folgende Teile vorliegen:

- HEXIO2 (Baugruppe mit Tasten 180mm*200mm)
- SBC3 (Steckbaugruppe 160mm*100mm)
- Steckernetzteil
- 2 Führungsschienen
- 2 Schrauben M3*8
- 2 Muttern M3
- 4 Christiani-Kurse "Mikroelektronik Einführung mit dem NDR-Computer"
- 1 Handbuch (in dem Sie gerade lesen)



Was es mit den einzelnen Baugruppen auf sich hat, erfahren Sie im nächsten Abschnitt.

Beschreibung der Bauelemente

HEXIO2

Die HEXIO2 ist die sog. Ein/Ausgabe-(englisch:Input/Output:I/O) Baugruppe Ihres Systems. Eine I/O-Baugruppe ist ein Gerät, mit dem Sie mit Ihrem Computer "sprechen" können und er mit Ihnen.

SBC3

Die SBC3 ist das Herz des Systems. Sie stellt den eigentlichen Computer dar. SBC steht für "Single-Board-Computer"; das bedeutet "Einplatinencomputer".

Steckernetzteil

Das Steckernetzteil ist die Spannungsversorgung des Computers. Es wird in eine Steckdose gesteckt und erzeugt dann eine Spannung, die Sie für den Betrieb Ihres Computers brauchen. Wenn es Sie interessiert: es sind +5,0 Volt Gleichspannung.
Bitte verwenden Sie nur das mitgelieferte Steckernetzteil. Bei der Verwendung eines Anderen besteht die Gefahr, daß Sie mit Ihrem Computer Rauchzeichen geben!

Führungsschienen

Die Führungsschienen sollen die SBC3 (den eigentlichen Computer) auf der Ein/Ausgabeeinhiet (HEXIO2) halten und mechanisch stabilisieren.

Und jetzt der Zusammenbau

Noch nichts einstecken: Bitte Geduld!

Nehmen Sie die HEXIO2-Baugruppe und schrauben Sie die beiden Führungsschienen an den dafür vorgesehenen Stellen an. Diese Stellen sind auf der Leiterplatte genauso gekennzeichnet, wie auf dem nebenstehenden Bild. Die benötigten Schrauben liegen dem System bei.

In die Führungsschienen, die Sie gerade angeschraubt haben, stecken Sie jetzt die SBC3 ein (**Bauteile vorne**). Durch die beiden Schienen können Sie dabei eigentlich nichts falsch machen. Achten Sie aber trotzdem darauf, daß jeder Stift der Stiftleiste der SBC3 in eine Buchse der Steckerleiste der HEXIO2 paßt. Sollte dies nicht der Fall sein, kontrollieren Sie bitte, ob einer der Stifte vielleicht verbogen oder geknickt ist. Gegebenenfalls müßten Sie ihn ge-

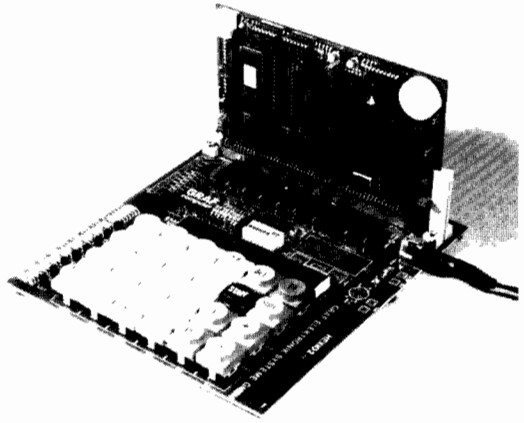
rade und parallel zu den anderen biegen. Jetzt müssen Sie nur noch für die nötige Spannung am System sorgen. Diese erzeugen Sie mit dem Steckernetzteil. Stecken Sie den kleinen Stecker wie auf dem folgenden Bild gezeigt, in die Kleinspannungsbuchse. Das Netzteil noch nicht in die Steckdose stecken!

Vor dem Einstecken: Platz bestimmen

Bevor Sie jetzt einstecken, sollten Sie erst einen geeigneten Platz für Ihren Computer finden. Ein geeigneter Platz sollte folgende Bedingungen erfüllen:

- trocken
- staubfrei
- keine zu heißen Unterlagen
- keine direkte Sonnenbestrahlung
- keine elektrisch leitenden Teile unter oder über dem System

Ideal wäre ein Tisch, auf dem Ihr System stehen bleiben könnte. Sollten Sie längere Zeit nicht damit arbeiten, wäre es sinnvoll Ihr System in einem Schrank zu deponieren, wo es vor Staub und äußeren Einwirkungen geschützt ist.



Nun Endlich: Einstecken

Achtung:

Verwenden Sie nur das mitgelieferte Steckernetzteil!

Bevor Sie jedoch das Steckernetzteil in eine Steckdose stecken, überprüfen Sie bitte nochmals Ihren Aufbau. Das fertig aufgebaute Gerät sollte genauso aussehen wie das System auf dem nebenstehenden Photo.

HALLO-1.1

Mit dem Einstecken des Steckernetzteils in die Steckdose, ist Ihr Computer eingeschaltet. Auf der Anzeige sollte nun eine Begrüßung Ihres Rechners an Sie stehen:

HALLO-1.1

Ist dies nicht der Fall drücken Sie bitte den RESET-Taster, den Sie links oben auf der SBC3 finden. Reagiert der Computer jetzt immer noch nicht, stecken Sie ihn aus und überprüfen Sie den Aufbau nocheinmal. Durch Druck auf den "RESET"-Taster kommen Sie immer wieder zum Anfangszustand. Es kann also nichts passieren!

Der erste Test: Es muß blinken

Mit Ihrem System haben Sie auch ein EPROM mitgeliefert bekommen, auf dem sich einige kleinere Programme befinden. Diese können Sie mit einigen einfachen Befehlen starten. Als erstes starten Sie einmal das Programm **"Blinklicht"**. Die Vorgehensweise ist hierbei, wie folgt:

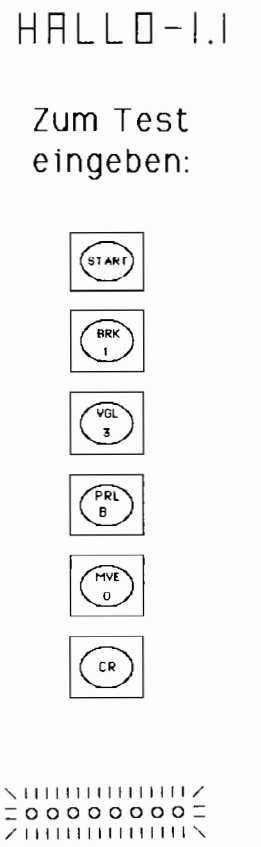
1. Drücken Sie nach dem Einschalten des Systems die Taste **START**. Auf der Anzeige Ihres Systems erscheint jetzt ein Vorschlag für eine mögliche Startadresse Ihres Programms (siehe Rand).

2. Da das Blinklichtprogramm aber auf einem anderen Speicherplatz abgelegt ist, müssen Sie diesen Platz dem Computer mitteilen. Ansonsten wählt er den angezeigten Platz als Anfangsadresse aus. Geben Sie jetzt nacheinander die Zahlen **1 3 B 0** ein. Dies ist die Startadresse des Blinklicht-Programms.

4. Jetzt müssen Sie dem Rechner nur noch sagen, daß die eben eingegebene Adresse als Startadresse verwendet werden soll. Hierzu drücken Sie die **CR**-Taste.

Sobald Sie diese Befehle eingegeben haben, fangen die Leuchtdioden auf der HEXIO2 zu blinken an.

Nun müssen die Leuchdioden auf der HEXIO2 in gleichmäßigen Zeitabständen blinken.



Jetzt aber: Lehrbriefe korrigieren

Wenn Ihnen das Blinken inzwischen zuviel geworden ist (Abschalten durch "RESET-Taster"), sollten Sie jetzt daran gehen die mitgelieferten Christiani Kurse zu korrigieren. Dies ist deshalb nötig, da diese Kurse noch für die alte Version des Einsteigerpaketes geschrieben sind, wir Ihnen aber die aktuellste Version bieten wollen.

Auf den nächsten Seiten sind einige Blätter abgedruckt, die Sie statt der Seiten in den beiliegenden Christiani Kursen abheften sollten. Es ist sowieso empfehlenswert, sich nach dem Durcharbeiten der vier Hefte die einzelnen Abschnitte nach Themen geordnet in einem Ordner abzulegen. Auf diese Weise erhalten Sie ein wertvolles Nachschlagewerk das Ihnen bei vielen Fragen die beim Arbeiten mit Ihrem System auftauchen, helfen kann. Bei den auszutauschenden Seiten handelt es sich um folgende:

Lehrbrief 1:

Seite	B5
Seite	B6
Seite	B7
Seite	B8
Seite	B17
Seite	B18

Lehrbreif 3:

Seite	B88
Seite	B89
Seite	B92
Seite	B93
Seite	B94
Seite	B95
Seite	B96
Seite	B100

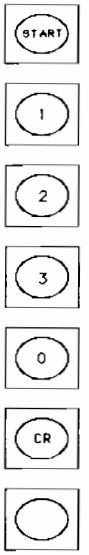
Lehrbrief 4:

Seite	A130
Seite	C21
Seite	D14

Die oben angesprochenen Blätter können Sie einfach gegen die im Kurs vorhandenen Seiten austauschen. Sie finden die Seiten im Anhang A an diese Beschreibung.

Zum Abschluß ein Spiel: Reaktionszeit

Genauso wie Sie vorher schon das Programm "Blinklicht" aufgerufen haben, können Sie jetzt auch das Programm "Reaktionszeit" aufrufen. Die Vorgehensweise ist hierbei genauso wie oben beschrieben (**START** drücken, Startadresse **1 2 3 0** eintippen und **CR** drücken).
Sie müssen statt der Adresse 1 3 B 0 von vorhin die Adresse **1 2 3 0** eingeben.

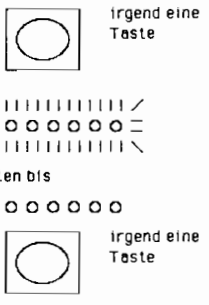


Wenn Sie das Programm starten, leuchten zunächst alle acht Leuchtdioden auf. Nach einer bestimmten Zeit, erlöschen die LED`s. Wenn Sie jetzt eine Taste drücken, mißt der Computer die Zeit, die Sie benötigt haben, um auf das Erlöschen der Leuchtdioden zu reagieren.

Die Zeit wird in Minuten/ Sekunden/ Zehntel und Hundertstel Sekunden angezeigt. Durch Druck auf eine beliebige Taste wird sie wieder angehalten.

Machen Sie 'mal einen Wettkampf:
00 00 15 ist schon ganz gut!

Natürlich können Sie diese Uhr auch als Stoppuhr verwenden!



Anschließend finden Sie eine Beschreibung und die Listings der Programme, die bereits fest in Ihrem System installiert sind.

Nichts geht mehr?

Für den Fall, daß sich Ihr System "aufhängen" sollte, gibt es eine recht einfache, aber sehr wirkungsvolle Methode, um den Normalzustand wieder herzustellen: Ein Druck auf den RESET-Taster!

Jetzt aber Christiani-Kurse durcharbeiten

Legen Sie jetzt dieses Handbuch beiseite und nehmen Sie die mitgelieferten Christiani-Kurse zur Hand. Arbeiten Sie die Kurse in aller Ruhe durch. Ein schnelles Überfliegen der einzelnen Abschnitte bringt nichts, weil Sie sich nichts einprägen können.

Los geht ` s mit dem Lernen

Auch beim Programmieren ist es wie bei anderen Tätigkeiten, wenn man es mit Freunden machen kann macht es doppelt soviel Spaß als wenn man es alleine tut. Es gibt bestimmt auch in Ihrer Bekanntschaft Leute, die sich für die Computerei interessieren. Außerdem ist es empfehlenswert, nicht die ganzen Kurse innerhalb von einem Tag durchzuarbeiten, da Sie nach einer bestimmten Zeit nicht mehr aufnahmefähig für den Lehrstoff sind. Lassen Sie sich ruhig Zeit; dann verstehen Sie den Inhalt des Lehrganges umso besser.

Abschließend noch einige Tips, damit Sie möglichst viel Freude mit Ihrem Computer haben:

- Schreiben Sie sich Ihre Programme immer auf, das wird Ihnen am Anfang zwar als sehr lästig erscheinen, aber spätestens wenn Sie das erste Programm geschrieben haben, das sich selbst zerstört, sind Sie vom Sinn dieser Maßnahme überzeugt.

- Die SBC3 verfügt über eine Akkupufferung die Ihre Programme auch nach dem Ausstecken des Systems im Speicher hält.

- Gewöhnen Sie sich an, Ihre Programme immer zu kontrollieren, um einer Selbstzerstörung des Programms vorzubeugen.

Genauere Einweisungen in die Programmierung in Z80-Maschinensprache geben Ihnen die beiliegenden "Christiani"-Kurse.

Nicht vergessen: Garantiekarte einsenden

Vergessen Sie bitte nicht Ihre ausgefüllte Garantiekarte an uns einzusenden. Nur wenn Sie das getan haben können wir eine Garantie für Ihr System übernehmen.

Empfehlenswertes Zubehör

Bestnr.:	Beschreibung:	Preis
10834	GES-Katalog	10,--
10286	Monitor-Listing	35,--
10061	LOOP-Abonnement	20,--
10096	18-polige Buchsenleiste (3 erforderlich)	3,80 pro Stück

Vorteile der Zeitschrift LOOP

Für ein Abonnement der Zeitschrift "LOOP" sprechen folgende Punkte:

- mit der Zeitschrift "LOOP" haben Sie ein wertvolles Nachschlagewerk für alles, was den NDR-Computer betrifft

- wenn Sie die LOOP abonniert haben sind Sie nicht alleine, da Sie dort Verbindungen mit anderen Computerbauern aufnehmen können

- Sie werden durch die LOOP auch immer auf dem laufenden gehalten, was Software für Ihr System angeht

- die LOOP ist immer aktuell, da sie alle zwei Monate neu erscheint

Viel Spaß beim Programmieren

Einige Programme sind schon fest 'drin. Falls Sie während des Durcharbeitens noch einige Programme ansehen wollen: Wir haben einige für Sie vorbereitet.

Sie können diese Programme zuerst 'mal starten und mit ihnen spielen. Wir wollen aber mehr:

Ihnen beibringen wie man "Programmiert"

Deshalb haben wir die Quellistings dieser Programme sowie umfangreiche Erklärungen beigefügt.

Hier noch die Programme mit Startadressen in einer kurzen Tabelle:

Adresse	Programmname
1 0 0 0 H	Lotto
1 0 5 0 H	Datum
1 1 5 0 H	Taste
1 1 7 0 H	Taste2
1 1 9 0 H	Lauf1
1 1 E 0 H	Lauf2
1 2 0 0 H	Lauf3
1 2 3 0 H	Reaktionszeittest
1 3 0 0 H	Würfel
1 3 6 0 H	Sägezahnfunktion
1 3 B 0 H	Blinklicht

Lottozahlen

Wenn Sie allwöchentlich das Problem haben, sich für Ihren Lotto-Tip zu entscheiden, dann kann Ihnen das Programm LOTTO helfen. Dieses Programm ermittelt mit Hilfe eines sogenannten Zufallszahlengenerators eine mögliche Tipfolge. Es kann natürlich nicht das richtige Ergebnis vorhersagen! Erwarten Sie also nicht sofort "Sechs Richtige" !

Wie kann der der Mikroprozessor "Zufallszahlen" erzeugen, er kann doch keinen Würfel werfen!? In der Tat ist der Begriff "Zufall" im Zusammenhang mit Computern umstritten. Schauen wir uns den erwähnten Würfel einmal genauer an: Wenn Sie alle Umstände genau kennen würden, Gewicht, genaue Form und Oberfläche des Würfels, evtl. Luftbewegungen, Richtung, Stärke und Drehimpuls des Wurfes etc., könnten Sie dann nicht das Ergebnis des Würfelns vorausberechnen? Genauso ist es bei computererzeugten Zufallszahlen. Wie fast alles in und um einen Computer ließen sich diese Zahlen vorhersagen, nur wäre das praktisch viel zu kompliziert und bei der Methode, die wir verwenden wollen, praktisch unmöglich.

Also noch einmal: Wie kann der Mikroprozessor Quasi-Zufallszahlen erzeugen?

Der Mikroprozessor Z80 bietet hier eine recht einfache Möglichkeit. Es gibt einen Befehl ld a,r (lade a mit dem Inhalt von r), wobei a der Akkumulator ist und r das "Refresh"-Register. Refresh heißt Auffrischen und bezeichnet bei speziellen Speicherbausteinen (dynamische RAMs) einen Vorgang, der regelmäßig wiederholt werden muß, um nicht den Speicherinhalt zu verlieren. Im Refresh-Register r des Z80 steht eine Zahl zwischen 0 und 127, die bei diesen Speicherbausteinen angibt, welcher Teil des Speichers als nächstes aufgefrischt werden muß. Da so ein Refresh sehr häufig durchgeführt werden muß, wird das Register r sehr schnell weitergezählt, so daß jedesmal, wenn wir einen Befehl ld a,r verwenden, im Akku eine andere Zahl ankommt.

Das Programm LOTTO benutzt genau diese Methode. Das Unterprogramm random dient hier dazu, eine Zufallszahl zwischen 1 und 38 zu erzeugen. Dazu wird zunächst der Akku mit dem Wert des Refresh Registers geladen, damit haben wir eine Zufallszahl zwischen 0 und 127. Nun wird von dieser Zahl solange 38 abgezogen, bis sie kleiner als 38 ist. Damit haben wir eine Zahl zwischen 0 und 37. Jetzt wird noch eine Eins zum Akkuinhalt addiert, das nennt man inkrementieren (englisch mit "c" statt "k", deshalb "inc"). Wir wollen die Zahl aber im Registerpaar HL haben, um sie einfach mit Hilfe des HEXMON-Unterprogrammes PrtDez (Print Dezimal) ausgeben zu können. Also laden wir das Register l mit dem Inhalt des Akkus und setzen das Register h auf Null. Soweit, so gut!
-Aber... Damit das LOTTO- Programm brauchbar wird, ist entprechend den Regeln die Einschränkung erforderlich, daß jede der Zahlenkugeln nur einmal gezogen werden darf. Eigentlich muß sich der Computer nur merken, welche Zahlen er bereits gezogen hat. Dafür gibt es mehrere Möglichkeiten, er könnte zum Beispiel eine Liste führen, in der alle Kugeln stehen, die noch nicht gezogen wurden. Oder er benutzt eine Tabelle, in der für jede Kugel steht, ob sie schon gezogen wurde. Nach diesem lezteren Verfahren arbeitet das Programm LOTTO.

Da für diese Methode bereits zu Beginn des Programmablaufes Vorbereitungen getroffen werden müssen, kehren wir also zurück an den Anfang des Programmes und erklären alles schön der Reihe nach (wie es sich schließlich für eine anständige Dokumentation gehört). Beim Start des Programms muß zunächst die oben erwähnte Tabelle erstellt werden. Das geschieht in der Schleife clrLoop. Zunächst wird aber die Anzahl der Kugeln in das Register b geladen, das wieder als Schleifenzähler dient. In das Registerpaar HL kommt die letzte Adresse, die noch zur Tabelle gehört, denn in clrLoop wird die Tabelle von hinten nach vorne erzeugt. Dazu wird in den Speicherplatz, dessen Adresse in HL steht, der Inhalt von b geschrieben. Dazu dient der Befehl ld (hl),b. HL enthält hier eine Adresse, man nennt das dann Zeiger (oder neudeutsch "pointer"), HL zeigt auf die Stelle in der Tabelle, an die die nächste Zahl kommt. Da die Tabelle von hohen zu niedrigen Adressen erstellt wird, muß HL nun um eins vermindert (dekrementiert) werden, das übernimmt der Befehl dec hl. Danach sorgt der djnz Befehl wieder dafür, daß die Schleife noch einmal durchlaufen wird, außer wenn b bereits den Wert Null erreicht hat. Anschließend steht an der Adresse store+1 eine 1, bei store+2 steht eine 2, bei store+10 steht eine 10 (0A sedezimal) u.s.w. Später soll überall dort, wo eine Kugel gezogen wurde eine Null stehen, damit das Unterprogramm random erkennen kann, ob die Zufallszahl, die es ermittelt hat noch vorhanden ist, oder ob eine andere ermittelt werden muß.

Nach dem die Tabelle eingerichtet ist folgt das Hauptprogramm. In einer Scheife, die maximal 7 mal durchlaufen wird, ruft das Hauptprogramm zunächst das Unterprogramm PRINT auf, das den Text "Lotto" zur Anzeige bereitstellt. Danach wird im Unterprogramm random die Zufallszahl generiert und auf Zulässigkeit hin überprüft. Wie wird's gemacht?

Dazu wird die Anfangsadresse der Tabelle, also die Adresse store, in das Registerpaar DE geladen (DE zeigt auf die Tabelle, HL ist der Index in diese Tabelle, also die laufende Nummer des Eintrags). Addiert man nun HL und DE, so erhält man die Adresse des Bytes, in dem eigentlich die Zufallszahl stehen müßte, das geschieht mit dem Befehl add hl,de.

HL zeigt jetzt also auf das fragliche Byte in der Tabelle. Dann vergleicht das Programm dieses Byte (also das Byte, dessen Adresse in HL steht) mit dem Akku (in dem immer noch die Zufallszahl steht): cp (hl). Ist der Unterschied Null, dann war das fragliche Byte dasselbe wie die Zufallszahl im Akku, das bedeutet, daß diese Kugel noch nicht gezogen wurde. Andernfalls, also wenn der Unterschied nicht Null ist, wurde sie schon einmal gezogen und der Befehl jr nz,random (Jump Relative if Not Zero) springt noch einmal an den Anfang des Unterprogramms, um eine neue Zahl zu erzeugen. (Das ist nicht unbedingt geschickt, aber da das Refresh-Register sich auch für den Mikroprozessor zu schnell ändert, kommt beim nächsten Versuch wahrscheinlich eine andere Zahl heraus. Praktisch merkt man nachher nicht, daß das Programm für einige Zahlen länger braucht als für andere.) Wenn nun (evtl. nach mehreren Versuchen) eine Zahl gezogen wurde, die noch nicht vergeben ist, muß diese Zahl jetzt als benutzt markiert werden. Dazu dient der Befehl ld (hl),0. Er lädt das Byte, dessen Adresse in HL steht mit dem Wert 0. Da das Hauptprogramm als Ergebnis des Unterprogramms eine Zahl im Registerpaar HL erwartet, nicht die Adresse, die jetzt noch dort steht, muß von HL wieder die Anfangsadresse der Tabelle, die immer noch in DE steht, abgezogen werden, dann kann die Rückkehr ins Hauptprogramm erfolgen. Leider gibt es den Befehl sub hl,de beim Z80 nicht, man darf hier aber auch sbc hl,de (SuBtract with Carry) verwenden, da an dieser Stelle im Programm kein Übertrag (Carry) vorliegen kann.
Abschliessend wird ins' Hauptprogramm zurückgesprungen (mit dem Befehl ret). Nach dem Befehl call HoleTaste folgt jetzt ein cp 57H. Warum das?
Nun, das Unterprogramm HoleTaste des HEXMON wartet bis eine Taste gedrückt wird und legt dann den Code dieser Taste im Akkumulator ab. Der Befehl cp (compare) vergleicht diesen Wert mit dem Code der CR Taste (57 sedezimal). Wurde die CR-Taste gedrückt, dann ist der Unterschied zum Vergleichswert Null. Deshalb springt der Befehl jr nz,ENDE (Jump Relative if Not Zero) zur Adresse ENDE wenn eine andere Taste als CR gedrückt wurde. Man kann dieses Programm also durch drücken einer beliebigen Taste abbrechen oder mit CR weitermachen.

Starten Sie dieses Programm mit dem START-Befehl, so erscheint die Schrift "Lotto" und die erste gezogene Zahl. Jetzt können Sie sich diese Zahl notieren und CR drücken um die nächste Zahl zu bekommen, oder Sie drücken irgendeine andere Taste, dann erlischt die Anzeige und nach einem kurzem Moment erscheint das HALLO-1.1. Das passiert auch, wenn Sie alle sieben Zahlen gezogen haben.

Lassen Sie sich jetzt einmal alle Zahlen anzeigen, das heißt Sie starten das Programm und drücken immer wieder CR bis das HALLO-1.1 erscheint. Nun können Sie sich die vom Programm angelegte Tabelle des Programms "angucken". Drücken Sie dazu die Taste SPE und geben als Adresse den sedezimalen Wert von store (=80D9 sedezimal) an. Dies ist der Eintrag für die Kugel Null, da es aber keine Null gibt, steht hier nichts sinnvolles. Also drücken Sie + oder CR um zum nächsten Speicherplatz zu kommen. Dort erscheint nun 01 oder 00, je nachdem ob die Zahl 1 gezogen wurde oder nicht. Schreiben Sie diesen und alle folgenden Speicherplätze auf, so erhalten Sie zum Beispiel die Folge:

01 02 03 04 05 06 07 00 00 0A 0B 0C 00
0E 0F 10 11 12 13 14 15 16 17 18 19 00
1B 1C 1D 1E 1F 00 00 22 00 24 25 26 und
hier ist die Tabelle zu Ende.
Da überall dort, wo eine Null steht eine Zahl gezogen wurde, muß das Programm folgende Zahlen ermittelt haben (sedezimal): 08 09 0D 1A 20 21 23.
Tatsächlich hatte es diese Zahlen in folgender Reihenfolge ausgegeben:
9 26 33 35 32 8 13 (dezimal).

Vielleicht benutzen Sie den hier vorgestellten Zufallszahlengenerator einmal als Grundlage für ein selbstentworfenes, kleines Würfelspiel ?


```
                ; Programm  LOTTO-1     1.0
                ;
                ; Dies Programm ermittelt per Zuffalszahlengenerator die Lottozahlen für
                ; die nächste Ziehung, leider arbeitet der Algorithmus noch nicht fehlerfrei,
                ; so daß die "Sechs Richtigen" nicht garantiert werden können.
                .Z80
                Ziehungen  EQU  6+1     ; Wieviel Kugeln werden gezogen
                Kugeln     EQU  38      ; aus wieviel Kugeln wird gezogen

                HoleTaste  EQU  000CH   ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                Print      EQU  0015H   ; Kopiert einen Text in die Anzeige
                PrtDez     EQU  0021H   ; Druckt eine Dezimalzahl
                Anzeige    EQU  8000H   ; Adresse des Anzeige-Speichers

                store      EQU  80FFH-Kugeln  ; Freier RAM-Speicher für eine Tabelle der Zahlen

                           ORG 1000H

                LOTTO:

                ; Zunächst wird eine Tabelle angelegt, in der steht, welche Zahlen noch nicht
                ; gezogen wurden, es werden praktisch alle Kugeln in die Ziehungsmaschine
                ; gefüllt. Diese Tabelle beginnt bei der Adresse store, sie muß im RAM stehen.

1000  06 26               ld b,Kugeln     ; wieviel Kugeln gibt es
1002  21 FF 80            ld hl,store+Kugeln  ; letztes Byte der Tabelle für noch vorhandene
                clrLoop:                  ; Zahlen, die hier angelegt wird
1005  70                  ld (hl),b       ; Zahl eintragen
1006  2B                  dec hl          ; Adresse herunterzählen
1007  10 FC               djnz clrLoop    ; für jede Kugel wiederholen

                ; Jetzt werden die Ziehungen durchgeführt.

1009  06 07               ld b,Ziehungen  ; dazu muß die Anzahl in Register b stehen
                loop:
100B  C5                  push bc         ; Zähler für die Wiederholung retten
100C  21 43 10            ld hl,text      ; Adresse des Textes in das Registerpaar HL laden
100F  CD 15 00            call Print      ; Unterprogramm zur Textausgabe aufrufen
1012  CD 29 10            call random     ; Unterprogramm zum würfeln einer Zahl
1015  DD 21 07 80         ld ix,Anzeige+7 ; Hier kommt die Zahl hin
1019  CD 21 00            call PrtDez     ; Gewürfelte Zahl ausgeben
101C  CD 0C 00            call HoleTaste  ; Auf Tastendruck warten
                                          ; Hier steht im Akku der Tastencode
101F  FE 57               cp 57H          ; Taste CR gedrückt ?
1021  20 03               jr  nz,ENDE     ; wenn nicht Programm beenden
1023  C1                  pop bc          ; Schleifenzähler wiederherstellen
1024  10 E5               djnz loop       ; b herunterzählen und springen falls nicht null
                ENDE:
1026  C3 00 00            jp  0000H       ; Zurück in den HEXMON springen,
                                          ; nach kurzer Zeit erscheint wieder das HALLO-1.1

                ; Soweit das Hauptprogramm,
                ; jetzt kommt nur noch das Unterprogramm zum Ziehen einer Zahl.

                random:                   ; Würfel eine Zahl von 1 bis Kugeln
1029  ED 5F               ld a,r          ; lade den Refresh-Wert in den Akku, dieser Wert ist
                                          ; zufällig genug
102B  FE 26               modulo: cp Kugeln       ; Ist der Wert kleiner als Kugeln ?
102D  38 04               jr c,endrandom  ; wenn ja fertig
102F  D6 26               sub Kugeln      ; sonst Kugeln vom Wert abziehen
1031  18 F8               jr modulo       ; und noch einmal testen
                endrandom:
1033  3C                  inc a           ; der Wert liegt zwischen 0 und Kugeln-1
```

```
                                        ; er soll aber zwischen 1 und Kugeln liegen, also
                                        ; wird er um eins erhöht (inkrementiert)
1034  6F                        ld l,a          ; das Ergebnis soll im Registerpaar HL stehen
1035  26 00                     ld h,00h        ; das obere Byte ist null
1037  11 D9 80                  ld de,store     ; Anfangsadresse der Tabelle in Registerpaar de laden
103A  19                        add hl,de       ; Adresse in der Tabelle berechnen
103B  BE                        cp (hl)         ; Vergleich die Zahl in der Tabelle mit der Zahl im
                                        ; Akkumulator, dort steht immer noch die gezogene
                                        ; Kugel, in der Tabelle steht entweder Null, oder
                                        ; auch die gezogene Kugel
103C  20 EB                     jr  nz,random   ; standen verschiedene Zahlen in Akku und Tabelle,
                                        ; dann wurde diese Zahl schon einmal gezogen.

                        ; Jetzt wurde eine neue Zahl gezogen

103E  36 00                     ld (hl),0       ; Diese Zahl löschen, damit sie nicht noch einmal
                                        ; gezogen werden kann
1040  ED 52                     sbc hl,de       ; Adresse der Tabelle wieder abziehen, damit die
                                        ; gezogene Zahl in HL steht
1042  C9                        ret             ; zurück ins Hauptprogramm


                        text:
1043  C7 A3 87 87 A3 FF FF FF   DB  0C7h, 0a3h, 87h, 87h, 0a3h, 0ffh, 0ffh, 0ffh
                                        ; hier steht  L o t t o
                        END
```


DATUM

Wissen Sie eigentlich schon, auf welchen Wochentag der Jahrtausendwechsel, also Sylvester 1999 fallen wird? Ich kann es Ihnen verraten: der 31.12.1999 wird ein Freitag sein. Um das herauszufinden, habe ich keinen "Tausendjährigen Kalender" oder ähnlich umständliche Sachen benötigt, mir hat ein Einsteigerpaket gereicht. Mit dem richtigen Programm kann das nämlich für beliebige Daten den Wochentag berechnen.

Das ist möglich, da unser Kalender, der sogenannte Gregorianische Kalender, den Papst Gregor der XIII. 1582 einführte, sehr regelmäßig ist. Bis 1582 galt der Julianische Kalender, in dem immer wieder Änderungen vorgenommen werden mußten, da die Sonne und der Mond sich nicht nach diesem Kalender richten wollten. Seit 1582 werden diese Änderungen durch die Schaltjahre vermieden. Diese werden nach einfachen Regeln ermittelt:

- Alle durch vier teilbaren Jahre sind Schaltjahre (zum Beispiel 1992 / 4 = 498, also ist 1992 ein Schaltjahr).

- Alle durch 100 teilbaren Jahre sind kein Schaltjahre, obwohl sie durch vier teilbar sind (Zum Beispiel das Jahr 1900).

- Alle durch 400 teilbaren Jahre sind Schaltjahr, obwohl sie durch 100 teilbar sind. (Zum Beispiel das Jahr 2000)

Damit wird der Kalender recht einfach berechenbar. Man berechnet folgende Funktion:
x:= (Tag + (2*Monat)+(((3*Monat)+3) DIV 5) + Jahr + (Jahr DIV 4) - (Jahr DIV 100) + (Jahr DIV 400) + 2) MOD 7

Dann ist x eine Zahl zwischen 0 und 6, wobei 0 Samstag bedeutet, 1 Sonntag u.s.w. Leider gilt diese Formel nicht für Januar und Februar, diese Monate werden als 13. bzw. 14. Monat des Vorjahres gerechnet.

Dabei sind DIV und MOD Operatoren für die Division mit Rest. Ein Beispiel: 13 geteilt durch 5 gibt 2 Rest 3. Das schreibt man in den meisten höheren Programmiersprachen so: 13 DIV 5 = 2

und 13 MOD 5 = 3, also DIV liefert das Ergebnis der Division (ganzzahlig) und MOD den Rest.

DATUM ist ein Programm, das nach der obigen Formel den Wochentag für beliebige Daten nach 1582 berechnen kann. Starten Sie es einmal! Es erscheint die leere Anzeige mit einer Null ganz rechts. Geben Sie nun den Tag ein, z.B. 13 und drücken dann CR. Wieder erscheint nur eine Null. Jetzt können Sie den Monat eingeben, z.B. 4 und die Eingabe mit CR abschließen. Schon wieder steht eine Null einsam in der Anzeige, jetzt fehlt noch die Eingabe des Jahres, z.B. 1965 und auch diese Eingabe wird mit CR beendet. Sofort erscheint ganz links die Schrift di, die Abkürzung für Dienstag. Und tatsächlich war der 13.4.1965 ein Dienstag. Jetzt haben Sie zwei Möglichkeiten: Drücken Sie noch einmal CR so können Sie einen weiteren Tag berechnen, drücken Sie irgendeine andere Taste landen Sie wieder im HEXMON.

Schauen wir uns das Programm mal näher an:

Es gliedert sich in drei große Teile und ein Unterprogramm. Recht einfach ist die Eingabe, da dazu nur Unterprogramme des HEXMON aufgerufen werden. Etwas schwieriger ist die Ausgabe, denn dort muß aus einer Zahl zwischen 0 und 6, die im Registerpaar BC steht, ein Text gemacht werden. Dazu wird einfach zu dieser Zahl die Startadresse einer entsprechenden Tabelle, die TagTabelle heißt, addiert. An dieser Adresse steht nun das erste Zeichen, das einfach in den Anzeigespeicher kopiert wird. Sieben Byte weiter in der Tabelle steht das zweite Zeichen, da es sieben Tage gibt. Also wird zu der Adresse noch einmal sieben addiert und das an dieser Adresse stehende Zeichen in die zweite Stelle des Anzeigespeichers kopiert.

Das eigentlich komplizierte ist aber die Berechnung der Formel. Hier muß mit 16-Bit Werten gerechnet werden, da z.B. die Jahreszahl nicht in einem Byte dargestellt werden kann. Deshalb wird das Programm etwas länger. Ausserdem gibt es keinen Z80 Befehl zum dividieren oder um den Rest zu berechnen. Dafür mußte ein Unterprogramm her, das dividiere heißt. Da hier dieses Unterprogramm nicht sehr oft aufgerufen wird, muß es auch nicht

besonders schnell sein. Es wird die "natürliche Methode" benutzt, solange die zu teilende Zahl größer oder gleich der Zahl, durch die geteilt wird ist, wird diese einfach von der anderen abgezogen. Nun muß nur noch mitgezählt werden, wie oft dies gelungen ist. Anders formuliert: Der Rechner zählt, wie oft er die Zahl, durch die geteilt wird, von der anderen abziehen kann. In unserm Beispiel vom Anfang: 13 durch 5 ist ? 13 ist größer als 5, also kann 5 von 13 abgezogen werden, das gibt 8. 8 ist größer als 5, also kann 5 von 8 abgezogen werden, das gibt 3. 3 ist kleiner als 5, also sind wir fertig. Wir konnten zweimal abziehen und es sind drei übriggeblieben, also 13 DIV 5 ist 2 und 13 MOD 5 ist 3. Auch in diesem Unterprogramm macht es sich unangenehm bemerkbar, daß wir mit 16-Bit Zahlen rechnen müssen, denn der Z80 hat keinen Vergleichsbefehl für solche Zahlen. Man muß also die einzelnen Bytes der beiden Zahlen miteinander vergleichen. Ist das höherwertige Byte der ersten Zahl kleiner als das der zweiten, dann muß die ganze erste Zahl kleiner sein, wir sind fertig. Sind die beiden höherwertigen Byte ungleich, dann ist also das erste größer, und damit ist die ganze erste Zahl größer. Es bleibt noch der Fall, daß die höherwertigen Byte gleich sind, dann müßen wir die niederwertigen Byte vergleichen. Können wir nicht weiter subtrahieren, dann muß das Ergebnis noch in die gewünschten Register gebracht werden. Hier fällt auf, daß der Z80 auch nicht alle denkbaren 16-Bit Transportbefehle kennt, deshalb wird hier ein Umweg über den Stack gemacht. Die Befehlsfolge push hl pop bc bewirkt dasselbe wie ld bc,hl.

Vielleicht an dieser Stelle mal eine Erklärung zum Stack:

Der Stack von dem hier die Rede ist, ist ein Top-Down-Stack (zu deutsch: Von-Oben-Nach-Unten-Stapel). Das bedeutet, daß die ersten Eintragungen in den oberen Regionen des Stapels abgelegt werden und man sich mit jeder weiteren Eintragung immer weiter nach unten bewegt. Diese Vorgehensweise erscheint uns zunächst etwas befremdend, wenn wir von der Vorstellung eines Bücherstapels ausgehen, denn dieser wächst ja von unten nach oben. Vielleicht hilft Ihnen auch folgendender Vergleich zum Verständnis des Top-Down-Stapels: Man

bezeichnet ihn mitunter auch als "Kellerstapel", weil die deutsche Übersetzung "Von-Oben-Nach-Unten-Stapel" doch etwas unbeholfen wirkt, und zum anderen der Stapel bei fortlaufender Beschickung von einer fiktiven Decke aus in den "Keller" herabwächst. Beiden Stapelarten ist jedoch gemeinsam, daß zuerst alle darüberliegenden (beim Bücherstapel) bzw. alle darunterliegenden (beim Top- Down- Stapel) Ablagen entfernt werden müssen, um an ein bestimmtes Objekt zu gelangen. In beiden Fällen nennt man das auch LIFO-Prinzip (Last In - First Out), da das letzabgelegte (Last In) Objekt in jedem Falle als Erstes (First Out) wieder vom Stapel genommen werden muß, um an eventuell höherliegende Objekte (denken Sie an den von der Decke aus herabwachsenden Stapel) gelangen zu können. Genauso arbeitet der Z80 mit seinem Stapel. Auf diesem Stapel liegen (natürlich) nur Zahlen. Diese Zahlen kommen aus den Registern des Prozessors mit dem Befehl "push..." und werden nachher auch wieder in Register gebracht mit dem Befehl: "pop...". Der Befehl push ix legt also die Zahl, die im Register IX steht auf den Stapel. Nun kann der Prozessor also nur noch an diese Zahl heran, da sie ganz "unten" im Stapel liegt. Bekommt er nun den Befehl pop hl, dann ist es ihm egal, aus welchem der Register die zu holende Zahl in den Stapel geschoben wurde, er nimmt sich lediglich die zuletzt abgelegte Zahl des Stapels und gibt sie in das HL-Register. So ist es also möglich, Daten von Registern in andere zu bringen, für die es keinen Transportbefehl (ld) gibt.


```
; Programm  DATUM     1.0
;
; Dieses Programm berechnet zu einem Datum den Wochentag

.Z80

Tag       EQU  80F0H  ; Speicher für den Tag
Monat     EQU  80F1H  ; Speicher für den Monat
Jahr      EQU  80F2H  ; Speicher für das Jahr (2 Byte)
x         EQU  80F4H  ; Hilfsvariable

HoleTaste EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
GetDez    EQU  002AH  ; Liest eine Dezimalzahl in HL ein
Clear     EQU  0033H  ; Löscht die Anzeige
Anzeige   EQU  8000H  ; Adresse des Anzeige-Speichers

          ORG  1050H

DATUM:

; Zunächst ein Datum einlesen

1050  DD 21 07 80       ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1054  21 00 00          ld hl,0         ; Default Wert in HL
1057  CD 2A 00          call GetDez     ; Dezimalzahl einlesen
105A  22 F0 80          ld (Tag),hl     ; Ergebnis speichern, das zweite Byte wird wieder
                                        ; überschrieben
105D  DD 21 07 80       ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
1061  21 00 00          ld hl,0         ; Default Wert in HL
1064  CD 2A 00          call GetDez     ; Dezimalzahl einlesen
1067  22 F1 80          ld (Monat),hl   ; Ergebnis speichern, das zweite Byte wird wieder
                                        ; überschrieben
106A  DD 21 07 80       ld ix,Anzeige+7 ; letzte Stelle der Anzeige steht in IX
106E  21 00 00          ld hl,0         ; Default Wert in HL
1071  CD 2A 00          call GetOez     ; Dezimalzahl einlesen
1074  22 F2 80          ld (Jahr),hl    ; Ergebnis speichern, diesmal sind zwei Byte gültig
1077  CD 33 00          call Clear      ; Anzeige löschen

; Dann daraus den Wochentag berechnen

107A  3A F1 80          ld a,(Monat)    ; Monat in den Akku laden
107D  FE 03             cp 3            ; ist es Januar oder Februar ?
107F  30 09             jr nc,MonOK     ; nein, Korrektur überspringen

   ; Korrektur für Januar und Februar, zur Berechnung der Schalttage

'81  C6 0C              add a,12        ; Monat um 12 erhöhen
,083  32 F1 80          ld (Monat),a    ; und wieder speichern
1086  21 F2 80          ld hl,Jahr      ; Adresse von Jahr in HL laden
1089  35                dec (hl)        ; um eins erniedrigen, zum Ausgleich, da Monat um
                                        ; 12 erhöht wurde
MonOK:
108A  2A F0 80          ld hl,(Tag)     ; den Tag in l laden, dabei kommt der Monat nach h
108D  26 00             ld h,0          ; h ist Null, HL enthält jetzt den Tag
108F  ED 5B F1 80       ld de,(Monat)   ; in E steht der Monat
1093  16 00             ld d,0          ; D ist Null, also steht in DE der Monat
1095  19                add hl,de       ; Monat zu HL addieren
1096  19                add hl,de       ; noch einmal addieren
1097  22 F4 80          ld (x),hl       ; und Ergebnis Speichern
109A  62                ld h,d          ; Monat nach HL laden
109B  6B                ld l,e
109C  19                add hl,de       ; Monat zu HL addieren
109D  19                add hl,de       ; noch einmal addieren, jetzt steht 3*Monat in HL
```

```
109E  11 03 00                ld de,3
10A1  19                      add hl,de       ; 3 dazu addieren
10A2  11 05 00                ld de,5         ; In DE steht 5
10A5  CD 1C 11                call dividieren ; Unterprogramm zum teilen, in HL steht jetzt
                                              ; ((3*Monat)+3)/5
10A8  ED 5B F4 80             ld de,(x)       ; Zwischenwert nach DE bringen
10AC  19                      add hl,de       ; zum bisherigen Zwischenwert addieren
10AD  ED 5B F2 80             ld de,(Jahr)    ; Jahr nach DE bringen
10B1  19                      add hl,de       ; und auch addieren
10B2  23                      inc hl          ; plus zwei,
10B3  23                      inc hl          ; also zweimal inkrementieren
10B4  22 F4 80                ld (x),hl       ; und wieder speichern
10B7  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10BA  11 04 00                ld de,4         ; durch 4 teilen,
10BD  CD 1C 11                call dividieren ;  gibt die Zahl der Schaltjahre
10C0  ED 5B F4 80             ld de,(x)       ; Zwischenwert holen
10C4  19                      add hl,de       ; zum addieren
10C5  22 F4 80                ld (x),hl       ; und wieder speichern
10C8  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10CB  11 64 00                ld de,100       ; durch 100 teilen,
10CE  CD 1C 11                call dividieren ;  gibt die Zahl der ausgefallenen Schaltjahre
10D1  54                      ld d,h          ; nach DE kopieren
10D2  5D                      ld e,l
10D3  2A F4 80                ld hl,(x)       ; Zwischenwert laden
10D6  ED 52                   sbc hl,de       ; ausgefallene Schaltjahre abziehen
10D8  22 F4 80                ld (x),hl       ; und wieder speichern
10DB  2A F2 80                ld hl,(Jahr)    ; Jahr in HL
10DE  11 90 01                ld de,400       ; durch 400 teilen,
10E1  CD 1C 11                call dividieren ;  gibt die Zahl der Schaltjahre, die nicht ausfielen
10E4  ED 5B F4 80             ld de,(x)       ; Zwischenwert laden
10E8  19                      add hl,de       ; zum addieren
10E9  11 07 00                ld de,7         ; durch 7 Teilen,
10EC  CD 1C 11                call dividieren ; der Rest ist die Nummer des Wochentags

                      ; Nun das Ergebnis ausgeben
                      ; In BC steht eine Zahl zwischen 0 und 6, dabei steht 0 für Samstag, 1 für
                      ; Sonntag, .... und 6 für Freitag

10EF  60                      ld h,b          ; Ergebnis nach HL bringen
10F0  69                      ld l,c          ; da dort gerechnet werden kann
10F1  11 0E 11                ld de,TagTabelle
10F4  19                      add hl,de       ; Adrsse des ersten Buchstabens in der Tabelle
                                              ; berechnen, steht in HL
10F5  7E                      ld a,(hl)       ; Buchstaben holen
10F6  32 00 80                ld (Anzeige),a  ; und in die Anzeige schreiben
10F9  11 07 00                ld de,7         ; jetzt Adresse des zweiten Buchstabens berechnen,
10FC  19                      add hl,de       ; der steht sieben Byte weiter
10FD  7E                      ld a,(hl)       ; zweiten Buchstaben holen
10FE  32 01 80                ld (Anzeige+1),a ; auch in die Anzeige schreiben
                      Fertig:
1101  CD 0C 00                call HoleTaste  ; Auf Tastendruck warten, damit die Anzeige gelesen
                                              ; werden kann
1104  FE 57                   cp 57h          ; Taste  CR  gedrückt ?
1106  20 03                   jr nz,ENDE      ; nein ? Dann Programm beenden.
1108  C3 50 10                jp DATUM        ; sonst noch einmal starten
110B  C3 00 00        ENDE:   jp 0            ; zum HEXMON-Start springen


110E  92 92 AA A1 AA A1 8E    TagTabelle:  DB     92h, 92h, 0aah, 0a1h, 0aah, 0a1h, 8eh
                              ; das heißt        S   S    M    d    M    d    F
1115  88 A3 A3 CF CF A3 AF                 DB     88h, 0a3h, 0a3h, 0cfh, 0cfh, 0a3h, 0afh
                              ; das heißt        A    o    o    I    I    o    r
```



```
                      dividieren:

                      ; Einfaches Unterprogramm zum Dividieren mit Rest
                      ; Für große Zahlen, die durch kleine geteilt werden, ist es nicht besonders
                      ; schnell, es gibt bessere Methoden.
                      ;
                      ; Eine Zahl steht in HL, sie wird durch DE geteilt.
                      ; Das Ergebnis steht wieder in HL, der Rest in BC

111C  F5                      push af         ; Akku und Flags auf den Stack retten
111D  DD E5                   push ix         ; ix auch retten
111F  DD 21 00 00             ld ix,0         ; In IX wird die Anzahl der Schleifendurchläufe
                                              ; mitgezählt
1123  7C              DivLoop: ld a,h          ; Oberes Byte des Divisors in A
1124  BA                      cp d            ; mit dem oberen Byte des Dividenten vergleichen
1125  38 0C                   jr c,DivEnde    ; Wenn kleiner muß gar nicht weiter getestet werden
1127  20 04                   jr nz,Div       ; HL ist größer als DE, also weitermachen
1129  7D                      ld a,l          ; Unteres Byte in A
112A  BB                      cp e            ; mit dem unteren Byte des Dividenten vergleichen
112B  38 06                   jr c,DivEnde    ; HL ist kleiner als DE, also fertig.
112D  DD 23           Div:    inc ix          ; Mitzählen
112F  ED 52                   sbc hl,de       ; einmal abziehen
1131  18 F0                   jr DivLoop      ; und wiederholen
                      DivEnde:
1133  E5                      push hl         ; Rest nach bc bringen
1134  C1                      pop bc
1135  DD E5                   push ix         ; Zahl der Schleifendurchläufe ist das Ergebnis
1137  E1                      pop hl
1138  DD E1                   pop ix          ; Register wieder herstellen
113A  F1                      pop af
113B  C9                      ret             ; und zurück zum Hauptprogramm

                      END
```

TASTE

Mit dem Programm TASTE können Sie auf einfache Art und Weise sehr schnell den Code einer Taste ermitteln. Wenn Sie zum Beispiel wissen wollen, welchen Code die Taste START erzeugt, starten Sie das Programm TASTE, darauf erscheint:

t A S t E

auf der Anzeige. Drücken Sie nun die START-Taste, so erscheint der Tastencode der START-Taste, nämlich 4b (sedezimal). Nun können Sie weitere Tastencode ermitteln, drücken Sie zum Beispiel CR, so erscheint 57, das ist der sedezimale Code der Taste CR. Vielleicht fällt Ihnen schon eine Regelmäßigkeit auf, probieren Sie doch mal BEF und SPE, dann bekommen Sie 47 und 5b. Schauen Sie sich nun die Anordnung der Tasten an und versuchen einmal zu erraten, welchen Code die 0- oder MVE-Taste hat! Um wieder einen HEXMON-Befehl eingeben zu können, müßen Sie die RESET Taste drücken, dann erscheint wieder das HALLO-1.1.

Wie macht das Programm das ?

Im Programm werden viele Unterprogramme des HEXMON benutzt:

HoleTaste

fragt solange die Tastatur ab, bis eine Taste gedrückt wurde. Während des Wartens ist die Anzeige sichtbar.

Print

kopiert einen Text (der hier Tabelle genannt wird) in die Anzeige. Dieser Text besteht aus acht Anzeigecoden, die man dem HEXMON-Handbuch entnehmen kann.

PrtAc

bedeutet print accumulator, es wandelt den Inhalt des Registers A (Akkumulator) in eine zweistellige Sedezimalzahl um und kopiert die Anzeigecode dieser Zahl an die Adresse, die in den Registern IX angegeben wird. Hier wird als Adresse der Speicherbereich für die Anzeige angegeben, so daß die Sedezimalzahl ganz links in der Anzeige erscheint. Macht man aus dieser Adresse Anzeige+6, bzw. aus dem Befehl 21 06 80 statt 21 00 80,

und läßt man den Befehl call Clear weg (oder ersetzt ihn durch NOP-Befehle, CD 33 00 wird zu 00 00 00), dann bleibt die Schrift tAStE in der Anzeige stehen und der Tastencode erscheint rechts daneben.

Clear

löscht den Speicher für die Anzeige. Außerdem wird noch die Adresse Anzeige (8000 sedezimal) verwendet, dort speichert HEXMON die Anzeige. (Das ist notwendig, da aus technischen Gründen (Multiplexbetrieb) immer nur eine der acht Anzeigeeinheiten arbeitet. Um nun eine 8-stellige Anzeige zu bekommen, muß HEXMON regelmäßig alle Anzeigen nacheinander einschalten. Dafür braucht er aber wiederum die Information, was angezeigt werden soll. Diese Information steht im Speicher von 8000 bis 8007 sedezimal. Drücken Sie die RESET-Taste, dann erscheint evtl. eine der Anzeigen heller, während alle anderen dunkel werden. Das liegt daran, das HEXMON nun nicht mehr alle Anzeigen nacheinander aktiviert, so daß nur die zuletzt eingeschaltete leuchtet.)

Das eigentliche Programm ist dann recht einfach: Mit Hilfe des Unterprogrammes Print wird der Text tAStE in die Anzeige geschrieben, dann wird mit Hilfe des Unterprogrammes HoleTaste ein Tastedruck erwartet. Der Tastencode dieser Taste steht anschließend im Akkumulator. Nun wird dieser Code mit Hilfe des Unterprogramms PrtAc in die Anzeige geschrieben, nachdem vorher mit dem Unterprogramm Clear die Anzeige gelöscht wurde. Das war's eigentlich. Um weitere Tastencode ermitteln zu können, folgt noch ein Sprung (jump) zur Stelle loop (Schleife), wo dann wieder auf einen Tastendruck gewartet wird.

Verwendete Befehle:

call		ruft ein Unterprogramm auf.
ld	load	lädt ein Register oder ein Registerpaar (z.B. HL) mit einem Wert, um Beispiel der Adresse "Tabelle"
jr	jump relative	springt zu einer anderen Stelle im Programm, wobei die Stelle nicht als absolute (sedezimal vierstellige) Adresse angegeben wird, sondern als

relative Adresse (auch Displacement oder Offset was soviel wie "Versatz" bedeutet). Das Displacement darf jedoch nur zwei Sedezimalstellen (entsprechend 1Byte) groß sein. Das hat zur Folge, daß ein Vorwärtssprung über maximal 127 Speicherzellen und ein Rückwärtssprung über maximal 128 Speicherstellen hinweg erfolgen kann. Bei der Interpretation des Displacements wird das Höchstwertigste der acht Bits als Vorzeichenbit aufgefaßt und die sieben niederwertigeren Bits ergeben nur noch einen Wertebereich von 127 (=2^6+ 2^5+ 2^4+ 2^3+ 2^2+ 2^1+ 1). Dieser Wert wird zur Adresse im Register PC (dem Befehlszähler) addiert. Dort stand nämlich bisher die Adresse, an der der jr- Befehl im Programm steht. Das Ziel des Sprunges bezieht sich also auf die Adresse des Sprungbefehls, daher der Name relative Adresse.


```
                        ; Programm TASTE        1.0
                        ;
                        ; Kleines Utility, ermöglicht das ermitteln von Tastencodes
                        ; und zeigt, wie man HEXMON Unterprogramme nutzen kann

                        .Z80

                        HoleTaste  EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        Print      EQU  0015H  ; Kopiert einen Text in die Anzeige
                        PrtAc      EQU  0018H  ; Schreibt den Akumulator-Inhalt Sedezimal in die Anzeige
                        Clear      EQU  0033H  ; Löscht die Anzeige
                        Anzeige    EQU  8000H  ; Adresse des Anzeige-Speichers

                                   ORG  01150H

1150  21 65 11          TASTE:  ld hl,tabelle   ; Adresse des Textes in Register HL
1153  CD 15 00                  call Print      ; Text ausgeben
1156  CD 0C 00          loop:   call HoleTaste  ; Auf Tastendruck warten, Tastencode steht im Acum.
1159  CD 33 00                  call Clear      ; Anzeigefeld löschen
115C  DD 21 00 80               ld ix,Anzeige+0 ; Position der Ausgabe in der Anzeige ganz links
1160  CD 18 00                  call PrtAc      ; Tastencode sedezimal ausgeben
1163  18 F1                     jr loop         ; wiederholen

                        tabelle:
1165  87 88 92 87 86 FF FF FF        db 87H, 88H, 92H, 87H, 86H, 0FFH, 0FFH, 0FFH
                        ; hier steht    t A S t E

                        end
```

```
                        ;  Programm TASTE-2        1.0
                        ;
                        ;  Kleines Utility, ermöglicht das ermitteln von Tastencodes
                        ;  und zeigt, wie man HEXMON Unterprogramme nutzen kann
                        ;  Es ist fast identisch mit dem Programm TASTE.
                        ;  Nur die Anzeige ist anders: der Text  tAStE bleibt ständig sichtbar
                        ;  und die Ausgabe der Tastenkodes erfolgt rechts daneben.

                        .Z80

                        HoleTaste  EQU  000CH  ; HEXMON Unterprogramm zum Abfragen des Tastenfeldes
                        Print      EQU  0015H  ; Kopiert einen Text in die Anzeige
                        PrtAc      EQU  0018H  ; Schreibt den Akumulator-Inhalt Sedezimal in die Anzeige
                        Anzeige    EQU  8000H  ; Adresse des Anzeige-Speichers

                                   ORG  01170H

1170  21 82 11            TASTE2: ld hl,tabelle   ; Adresse des Textes in Register HL
1173  CD 15 00                    call Print      ; Text ausgeben
1176  CD 0C 00            loop:   call HoleTaste  ; Auf Tastendruck warten, Tastencode steht im Acum.
1179  DD 21 06 80                 ld ix,Anzeige+6 ; Position der Ausgabe in der Anzeige ganz links
117D  CD 18 00                    call PrtAc      ; Tastencode sedezimal ausgeben
1180  18 F4                       jr loop         ; wiederholen

                        tabelle:
1182  87 88 92 87 86 FF FF FF        db 87H, 88H, 92H, 87H, 86H, 0FFH, 0FFH, 0FFH
                        ; hier steht    t A S t E

                        end
```


Lauflicht

Das Einsteigerpaket besitzt neben den Siebensegment-Anzeigen auch noch acht LEDs, die normalerweise alle leuchten. Gerade diese LEDs sind aber besonders einfach anzusteuern, da sie direkt mit dem IO-Port (Input/Output - Ein/Ausgabe) mit der Nummer 2 angeschlossen sind. Um zum Beispiel alle LEDs auszuschalten kann man den HEXMON Befehl IOS (Input/Output setzen) benutzen:

Wenn HEXMON sich mit "HALLO-1.1" oder "--------" oder "-bef-" meldet, drückt man die Taste 8/IOS. Daraufhin erscheint 00, damit ist die Nummer des IO-Ports gemeint. Sie wollen Port 2 setzen, also tippen Sie 2 CR, und erhalten die Anzeige 02 00. Nun können Sie den Wert, den Sie ausgeben wollen, angeben. Geben Sie einmal F F CR ein: die LEDs erlöschen. Tippen Sie nun 0 0 CR so leuchten alle wieder auf.

Wird auf diesen Port eine Zahl ausgegeben, so läßt jedes 0-Bit eine LED leuchten, ein 1-Bit dagegen schaltet sie aus. Probieren Sie einmal aus was passiert, wenn Sie nacheinander die Zahlen (sedezimal) FE, FD, FB, F7, EF, DF, BF und 7F auf Port 2 ausgeben! Nacheinander leuchten alle acht LEDs auf, wenn Sie schnell genug tippen könnten, würde der Eindruck eines Laufenden Punktes entstehen. Was liegt nun näher, als diese "Tippaufgabe" einem Programm zu übergeben? Dazu müssen Sie nur wissen, wie ein Programm einen Wert an einen Port ausgeben kann. Das geht recht einfach mit zwei Befehlen: ld a,0FFh (3E FF) und out (02H),a (D3 02) schaltet zum Beispiel alle LEDs aus.

Nun ein erstes Programm: Sie geben acht mal hintereinander diese vier Byte ein (mit dem HEXMON Befehel SPEeichern) und ersetzen dabei jedesmal die FF durch die weiter oben angegebene Folge FE, FD, FB, F7, EF, DF, BF und 7F. Also zum Beispiel an Adresse 8300 (sedezimal) 3E FE D3 02 3E FD D3 02 3E FB D3 02 3E F7 D3 02 3E EF D3 02 3E DF D3 02 3E BF D3 02 3E 7F D3 02 eigeben. Dann drücken Sie BEF und anschließend STEP. Geben Sie die erste Adresse an (im Beispiel 8300) und drücken CR. Jetzt müßte 8300 3E auf der Anzeige stehen. Wenn Sie jetzt mehrmals hintereinander CR drücken, wandert ein Lichtpunkt über die LEDs. Wenn die

LED ganz links leuchtet müssen Sie aufhören, denn unser Programm geht hier nicht weiter. Das können Sie aber leicht ändern: Fügen Sie noch zwei Byte an das Programm an, 18 und DE (im Beispiel auf Adresse 8320 und 8321); das bedeutet jr 8300 - jump relative to 8300, das Programm fängt wieder von vorne an. Probieren Sie das Programm nocheinmal mit dem STEP Befehl aus! Diesmal haben Sie schon ein komplettes Lauflicht. Wenn das Programm fehlerfrei arbeitet, können Sie daran gehen, den Programmablauf im Prozessortakt stattfinden zu lassen. Drücken Sie dazu BEF, START, geben die Adresse (im Beispiel 8300) ein und drücken CR. Nanu? Alle LEDs leuchten halbhell, irgendwas scheint nicht zu funktionieren! Drücken Sie erstmal RESET, damit das HALLO-1.1 wieder erscheint. Wahrscheinlich wissen Sie schon, wohin der Hase läuft: Das Programm ist zu schnell, unser Auge kann nicht folgen. Dann machen wir es doch langsamer! Nach jedem Schritt soll das Programm einen kleinen Moment warten. Warten ist etwas, das der Mikroprozessor eigentlich gar nicht kann, wir müssen ihn irgendend was tun lassen, er kann zum Beispiel zählen. Das können Sie in einem Unterprogramm programmieren, das dann nach jedem Schritt mit dem CALL Befehl aufgerufen wird. In diesem Unterprogramm lassen Sie dem Mikroprozessor zum Beispiel von B000 (sedezimal) bis FFFF (sedezimal) zählen und dann noch einmal von FFFF bis 0000 (sedezimal). Dabei tritt ein Übertrag auf (FFFF + 0001 ist ja eigentlich 10000). Diesen Übertrag (Carry) können Sie zum Abbruch des Wartens benutzen. Ihr Unterprogramm muß dann nur noch mit Hilfe des RET (Return) Befehls ins Hauptprogramm zurückkehren. Setzen Sie das alles zusammen, so erhalten Sie das Programm LAUF-1.

Das ist aber nur ein erster Ansatz, wir haben zwar ein funktionsfähiges Lauflicht, das Programm ist aber recht lang und etwas umständlich. Denn im Befehlssatz des Mikroprozessors Z80 gibt es eine Gruppe von Befehlen, die uns hier sehr helfen können: Die Rotations- und Schiebebefehle. Natürlich dienen diese Befehle nicht eigentlich dazu Lauflichter zu programmieren, man kann mit diesen Befehlen zum Beispiel in bestimmten Fällen Zahlen multiplizieren und dividieren.

Um die Wirkungsweise dieser Befehle zu verstehen, schauen wir uns zum Beispiel den RLCA Befehl an. RLCA bedeutet Rotiere Links Circulär den Akkumulator. Gemeint ist, daß Irgendetwas linksherum im Kreis bewegt wird, und daß dieses Etwas im Akkumulator steht. Im Akkummulator stehen doch Zahlen, wie kann man Zahlen denn im Kreis bewegen ? Schauen Sie sich folgendes Beispiel mit vierstelligen Dezimalzahlen an: 1234 -> 2341 -> 3412 -> 4123 -> 1234 uund so weiter. Das ist also mit "Rotiere Circulär" gemeint. Im Akkumulator des Mikroprozessors stehen aber keine Dezimal- sondern Binärzahlen. Wie sieht es damit aus ?

Auch dazu ein Beispiel:
0001 -> 0010 -> 0100 -> 1000 -> 0001
u.s.w.,
oder wie beim Z80 mit achtstelligen Binärzahlen:
00000001 -> 00000010 -> 00000100 ->
00001000 -> 00010000 -> 00100000 ->
01000000 -> 10000000 -> 00000001
u.s.w.

Wenn wir jetzt als erste Zahl 11111110 nehmen, dann entsteht genau die Folge, die wir oben für unser Lauflicht benutzt haben. Nach diesem Verfahren ist das Programm LAUF-2 geschrieben, nur wird hier als Startwert 11101110 verwendet, so daß immer zwei LEDs gleichzeitig leuchten. Dieser Wert wird zunächst in den Akkumulator geladen, dann folgt die Hauptschleife des Programms: Ausgeben des Akkumulators an die LEDs, warten, Akku zirkulär rotieren und das ganze wiederholen. Zum warten wird wieder das Unterprogramm wait benutzt, das Sie schon aus LAUF-1 kennen.

Dieses neue Verfahren läßt sich nun sehr leicht abwandeln. Sie können zum Beispiel 00000001 als Startwert nehmen (01 sedezimal), dann leuchten alle LEDs bis auf eine. Oder Sie nehmen den Wert 55 sedezimal, das ist 01010101 binär, dann wechseln sich jeweils eine leuchtende und eine dunkle LED ab.

Wenn Sie aber den Wert 6D sedezimal (01101101 binär) nehmen, sieht das Lauflicht etwas seltsam aus, das liegt daran, das Sie drei helle LEDs nicht gleichmäßig über 8-Bit verteilen können. Wir brauchen also ein neuntes Bit. Dieses neunte Bit kann man im Carry-Flag speichern. Das Carry-Flag ist ein Flag

(eine Flagge, entweder gesetzt oder nicht gesetzt, halbmast gibt es hier nicht), das einen Übertrag anzeigt (engl.: carry - Übertrag).

Dieses Carry-Flag wird von dem Befehl RLA, Rotiere den Akkumulator links herum, benutzt. Bei diesem Befehl wandert das Bit 7 des Akkumulators (das Bit ganz links) in das Carry, während das Carry in Bit 0 (das Bit ganz rechts) wandert:

	I	Carry	I	Akkuinhalt
1.	I	1	I	01101101
2.	I	0	I	11011011
3.	I	1	I	10110110
4.	I	1	I	01101101
5.	I	0	I	11011011
6.	I	1	I	10110110
7.	I	1	I	01101101
8.	I	0	I	11011011
9.	I	1	I	10110110

Das Carry kann nicht mit an die LEDs ausgegeben werden, da nur acht da sind. Trotzdem sieht das Lauflicht mit Programm LAUF-3 besser aus, als LAUF-2 mit dem Anfangsert 01101101 binär. Vergleicht man LAUF-2 mit LAUF-3, dann sieht man, das sich nicht nur der Rotierbefehl geändert hat. Beim Programmstart muß das Carry gesetzt werden, das macht der Befehl SCF - Set Carry Flag. Außerdem verändert das Unterprogramm wait das Carry-Flag, denn es zählt bis ein Übertrag auftritt. Wir wollen aber nach Aufruf des Unterprogramms unser Original-Carry wieder verwenden. Deshalb muß der Inhalt des Flag-Registers am Anfang des Unterprogrammes gerettet werden, das übernimmt der Befehl push AF (Schiebe Akku und Flags auf den Stack). Am Ende des Unterprogramms, vor dem Return zum Hauptprogramm, wird der alte Zustand wieder hergestellt mit dem Befehl pop AF (Nehme Akku und Flags vom Stack). Diese Befehle arbeiten mit einem besonderem Speicherbereich, dem Stack (Stapel). Wo dieser Speicherbereich anfängt steht im Register SP, dem Stackpointer (Zeiger auf den Stapel). Mit push Befehlen kann man Register des Z80 auf den Stack schieben, von dem man sie später mit pop Befehlen wieder herunterholt. Auf diese Art konnten wir den Inhalt des Akkumulators und des Carry-Flags zwischenspeichern, so daß das Hauptprogramm gar nicht merkt, daß diese Register im Unterprogramm benutzt wurden.


```
1190  3E FE
1192  D3 02
1194  CD CA 11
1197  3E FD
1199  D3 02
1198  CD CA 11
119E  3E FB
11A0  D3 02
11A2  CD CA 11
11A5  3E F7
11A7  D3 02
11A9  CD CA 11
11AC  3E EF
11AE  D3 02
11B0  CD CA 11
11B3  3E DF
11B5  D3 02
11B7  CD CA 11
11BA  3E BF
11BC  D3 02
11BE  CD CA 11
11C1  3E 7F
11C3  D3 02
11C5  CD CA 11
11C8  18 C6

11CA  21 00 B0
11CD  11 01 00

11D0  19
11D1  30 FD

11D3  C9
```

```
;  Programm  LAUF-1     1.0
;
;  Einfaches Lauflicht.

.Z80
        ORG 01190H

LAUF1:
        ld a,11111110b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die erste LED leuchtet
        call wait
        ld a,11111101b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die zweite LED leuchtet
        call wait
        ld a,11111011b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die dritte LED leuchtet
        call wait
        ld a,11110111b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die vierte LED leuchtet
        call wait
        ld a,11101111b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die fünfte LED leuchtet
        call wait
        ld a,11011111b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die sechste LED leuchtet
        call wait
        ld a,10111111b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die siebte LED leuchtet
        call wait
        ld a,01111111b  ; Akku laden,
        out (02H),a     ; Akku an die LEDs ausgeben, die achte LED leuchtet
        call wait
        jr LAUF1

wait:   ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
        ld de,0001H     ; oder hier
Wloop:
        add hl,de       ; HL um einen hochzählen (in DE steht 1)
        jr nc,Wloop     ; Wiederholen, bis Überlauf (Carry),
                        ; also springen, wenn kein Carry (no carry - nc)
        ret             ; zurück zum Hauptprogramm
end
```

```
                        ;   Programm  LAUF-2     1.0
                        ;
                        ;   Einfaches Lauflicht, aber geschickter programmiert als LAUF-1

                        .Z80
                               ORG 011E0H

                        LAUF2:
11E0  3E EE                    ld a,11101110b  ; Akku laden
                        LauLoop:
11E2  D3 02                    out (02H),a     ; Akku an die LEDs ausgeben, zwei LEDs leuchten
11E4  CD EA 11                 call wait2      ; warten
11E7  07                       rlca            ; Rotiere den Akku circulär nach links
11E8  18 F8                    jr LauLoop      ; Wiederholen

11EA  21 00 B0          wait2: ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
11ED  11 01 00                 ld de,0001H     ; oder hier
                        W2loop:
11F0  19                       add hl,de       ; HL um einen hochzählen (in DE steht 1)
11F1  30 FD                    jr nc,W2loop    ; Wiederholen, bis Überlauf (Carry),
                                               ; also springen, wenn kein Carry (no carry - nc)
11F3  C9                       ret             ; zurück zum Hauptprogramm
                        end
```



```
                        ;   Programm  LAUF-3     1.0
                        ;
                        ;   Lauflicht mit neun Takten

                        .Z80

                               ORG 01200H

                        LAUF3:
1200  3E 6D                    ld a,01101101b  ; Akku laden, drei LEDs leuchten
1202  37                       scf             ; Set Carry Flag, da Carry mitrotiert wird
                        LauLoop:
1203  D3 02                    out (02H),a     ; Akku an die LEDs ausgeben
1205  CD 0B 12                 call wait3      ; warten
1208  17                       rla             ; Rotiere den Akku nach links durch das Carry
1209  18 F8                    jr LauLoop      ; Wiederholen

1208  F5              wait3:   push af         ; rette Akku und Flags auf den Stack
120C  21 00 80                 ld hl,0b000H    ; Hier kann man die Geschwindigkeit einstellen
120F  11 01 00                 ld de,0001H     ; oder hier
                        W3loop:
1212  19                       add hl,de       ; HL um einen hochzählen (in DE steht 1)
1213  30 FO                    jr nc,W3loop    ; Wiederholen, bis Überlauf (Carry),
                                               ; also springen, wenn kein Carry (no carry - nc)
1215  F1                       pop af          ; Akku und Carry wieder in Originalzustand versetzen
1216  C9                       ret             ; zurück zum Hauptprogramm
                        end
```

REAKTIONSZEIT

Mit dem Programm Reaktionszeit können Sie feststellen, wie gut Ihre Reaktion auf ein Signal ist. In diesem Fall, müssen Sie auf ein Erlöschen der Leuchtdioden reagieren.

Das Programm wartet nach dem Aufruf zunächst auf eine Taste. Wenn Sie nun eine beliebige Taste drücken, beginnt das Programm zu laufen. Zunächst leuchten die acht Leuchtdioden auf der HEXIO2. Die Dauer diese Leuchtens ist bei jedem Durchlauf verschieden. Dies wird erreicht durch das Laden eines Zeitfaktors, mit einem Wert aus dem Refresh-Register. Das Refresh-Register ist ein Zähler, der die Speicher des Rechners nachläd. Das Nachladen der Register erfolgt in sehr kurzen Zeitabständen, da der Rechner sonst die Daten wieder vergißt.

Nach einer bestimmten Zeit erlöschen die LED's. Jetzt beginnt die Zeitmessung. In der Anzeige sehen Sie nun drei Blöcke von Zahlen. Der ganz links gelegene Block zählt die Minuten (solange sollten Sie wohl nicht brauchen). Der mittlere Block zählt die Sekunden die vergehen, ehe Sie eine Taste drücken. Der rechte Block schließlich, zählt die Zehntel und die Hundertstel Sekunden..

Durch ein erneutes Drücken, auf irgendeine Taste startet das Programm von neuem.


```
                .z80
                        TITLE   REAKTIONSTEST

HOLETASTE       EQU     000CH
CLEAR   EQU     0033H
ANZEIGE EQU     0009H
PRTAC   EQU     0018H
ANZFELD EQU     8000H
MINUTEN EQU     80FCH
SEKUNDEN EQU    80FDH
MILLISEKUNDEN EQU   80FEH

                        ORG     1230H

START:
1230 CD 33 00           CALL    CLEAR           ;Anzeige löschen

BEGINN:
1233 CD 0C 00           CALL    HOLETASTE       ;Tastatur abfragen

START1:
1236 3E 00              LD      A,00000000B
1238 D3 02              OUT     (02H),A         ;Akkuinhalt auf
                                                ;LED's ausgeben
123A 32 FC 80           LD      (MINUTEN),A     ;Anfangswerte
123D 32 FD 80           LD      (SEKUNDEN),A    ;einstellen
1240 32 FE 80           LD      (MILLISEKUNDEN),A
1243 CD B3 12           CALL    WAIT

LED$AUS:
1246 3E FF              LD      A,11111111B
1248 D3 02              OUT     (02H),A         ;LED's ausschalten

124A FD 21 FC 80        LD      IY,MINUTEN      ;Anfang der Zeit-
124E FD 36 02 00        LD      (IY+2),0        ;messung

LOOP:
1252 06 63              LD      B,063H          ;Anfang Warte-
                                                ;schleife
MILLISEK:
1254 CD 7F 12           CALL    ZEITANZEIGE
1257 10 FB              DJNZ    MILLISEK

1259 FD 7E 01           LD      A,(IY+1)        ;Sek.:=Sek.+1
125C 3C                 INC     A
125D 27                 DAA
125E FD 77 01           LD      (IY+1),A

1261 FE 60              CP      60H             ;Wenn Sek.=60
1263 20 14              JR      NZ,MILLIO
1265 FD 7E 00           LD      A,(IY+0)        ; Min.:=Min.+1
1268 3C                 INC     A
1269 27                 DAA
126A FD 77 00           LD      (IY+0),A

126D FE 60              CP      60H
126F 20 04              JR      NZ,SEKO
1271 FD 36 00 00        LD      (IY+0),0        ;Minuten:=0

SEKO:
1275 FD 36 01 00        LD      (IY+1),0        ;Sekunden:=0
```

```
                        MILLIO:
1279 FD 36 02 00                LD      (IY+2),0        ;Millisek.:=0

127D 18 D3                      JR      LOOP            ;unendlich viele
                                                        ;Durchläufe
                        ZEITANZEIGE:
127F C5                         PUSH    BC              ;Zähler einstellen
1280 06 03                      LD      B,03H

                        DISPLAY:
1282 DD 21 00 80                LD      IX,ANZFELD      ;Minuten anzeigen
1286 FD 7E 00                   LD      A,(IY+0)
1289 CD A8 12                   CALL    ANZEIGEN

128C DD 23                      INC     IX
128E FD 7E 01                   LD      A,(IY+1)        ;Sekunden anzeigen
1291 CD A8 12                   CALL    ANZETGEN

1294 DD 23                      INC     IX
1296 FD 7E 02                   LD      A,(IY+2)        ;Millisek. anzeigen
1299 CD A8 12                   CALL    ANZEIGEN

129C 10 E4                      DJNZ    DISPLAY         ;Zähler dekrementieren
                                                        ;bis Zähler=0
129E FD 7E 02                   LD      A,(IY+2)        ;Milli.:=Milli.+1
12A1 3C                         INC     A
12A2 27                         DAA
12A3 FD 77 02                   LD      (IY+2),A

12A6 C1                         POP     BC

12A7 C9                         RET

                        ANZEIGEN:
12A8 CD 18 00                   CALL    PRTAC           ;Akkuinhalt ausgeben

                        HOLL:
12AB CD 09 00                   CALL    ANZEIGE         ;Anzeige aus Grund-
                                                        ;programm aufrufen
                                                        ;gleichzeigig Tastatur
                                                        ;abfragen
12AE FE FF                      CP      OFFH            ;Taste gedrückt ?
12B0 38 14                      JR      C,STOP          ;Ja, Uhr anhalten

12B2 C9                         RET

                        WAIT:
12B3 ED 5F                      LD      A,R             ;Warteschleife
12B5 47                         LD      B,A             ;einstellen
                        W1:
12B6 21 00 F0                   LD      HL,0F000H
12B9 11 01 00                   LD      DE,0001H

                        W2:
12BC CD C2 12                   CALL    WAIT1
12BF 10 F5                      DJNZ    W1              ;B dekrementieren
                                                        ;bis B=00h
12C1 C9                         RET

                        WAIT1:
12C2 19                         ADD     HL,DE           ;HL und DE addieren
12C3 30 FD                      JR      NC,WAIT1        ;bis HL=0FFFFH
12C5 C9                         RET
```



```
                        STOP:
12C6 D1                         POP     DE
12C7 CD D5 12                   CALL    SCHLEIFE        ;Warteschleife aufrufen

                        LOOP2:
12CA CD 09 00                   CALL    ANZEIGE         ;Anzeigenroutine aufrufen

12CD FE FF                      CP      0FFH            ;Vergleich ob Taste gedrückt
12CF DA 33 12                   JP      C,BEGINN        ;wenn ja zum Anfang springen
12D2 18 F6                      JR      LOOP2

12D4 C9                         RET

                        SCHLEIFE:
12D5 21 01 00                   LD      HL,0001H        ;Warteschleife starten

                        SCHL1:
12D8 29                         ADD     HL,HL
12D9 30 FD                      JR      NC,SCHL1

12DB C9                         RET

                        ENDE:
                                END
```

WÜRFEL

Bei diesem Programm handelt es sich um einen elektronischen Würfel. Es ist aber nicht so, daß der Computer irgendwelche Zufallszahlen berechnet, sondern er durchläuft eine Schleife. In dieser Schleife, schaltet er nacheinander alle LED's ein.

Da er nach jeder LED, die er eigeschaltet hat, in eine Warteschleife springt, in der er die Tastatur abfragt. Wenn eine Taste gedrückt wird, bleibt der Rechner an dieser Stelle stehen und zeigt solange das Würfelergebnis an.


```
                        TITLE   WÜRFEL   3

holtaste        equ     000ch
clear   equ     0033h
tonum   equ     000fh
anzeige equ     0009h

                org     1300h

start:
1300  CD 33 00            call    clear           ;Anzeige löschen

auswahl:
1303  CD 0C 00            call    holtaste        ;Tastatur abfragen
1306  CD 0F 00            call    tonum           ;Tastencode bilden
1309  4F                  ld      c,a             ;und ins Register c laden

aus:
130A  3E 0F               ld      a,0fh           ;Akku mit 0fh laden
130C  B9                  cp      c               ;mit c vergleichen
130D  D4 13 13            call    nc,tab          ;wenn a=c dann Tabelle anspringen

1310  C3 03 13            jp      auswahl         ;sonst nocheinmal

tab:
1313  3E FD               ld      a,11111101b     ;Akku mit Wert laden
1315  D3 02               out     (02h),a         ;ausgeben auf die LED's
1317  CD 3F 13            call    wait            ;Warteschleife aufrufen

131A  3E F9               ld      a,11111001b     ;nächster Wert
131C  D3 02               out     (02h),a
131E  CD 3F 13            call    wait

1321  3E F1               ld      a,11110001b
1323  D3 02               out     (02h),a
1325  CD 3F 13            call    wait

1328  3E E1               ld      a,11100001b
132A  D3 02               out     (02h),a
132C  CD 3F 13            call    wait

132F  3E C1               ld      a,11000001b
1331  D3 02               out     (02h),a
1333  CD 3F 13            call    wait

1336  3E 81               ld      a,10000001b
1338  D3 02               out     (02h),a
133A  CD 3F 13            call    wait

133D  18 D4               jr      tab

wait:
133F  21 FF FF            ld      hl,0ffffh       ;hl mit 0ffffh laden

loop:
1342  29                  add     hl,hl           ;hl zu sich selbst addieren
1343  30 FD               jr      nc,loop         ;bis hl voll (0ffffh),d.h.hier ein
                                                  ;Durchlauf
1345  CD 52 13            call    holetaste       ;Tastatur abfragen
1348  CD 0F 00            call    tonum           ;Tastencode bilden
134B  4F                  ld      c,a             ;in c-Register laden
134C  3E 01               ld      a,01h           ;Akku mit 01h laden
134E  B9                  cp      c               ;mit c vergleichen
```

```
134F  30 B2              jr     nc,auswahl        ;Vergleich o.k.zum Anfang springen

1351  C9                 ret

                   holetaste:
1352  06 08              ld     b,8               ;Tastatur 8 mal abfragen

                   holl1:
1354  CD 09 00           call   anzeige           ;wenn keine Taste gedrückt
1357  FE FF              cp     0ffh              ;wurde wieder zu Hauptprogramm
1359  20 F7              jr     nz,holetaste      ;springen
135B  10 F7              djnz   holl1

135D  C9                 ret
```


SÄGEZAHN

Dieses Programm schaltet die LED's nacheinander ein. Wenn alle Leuchtdioden leuchten, werden sie im nächsten Schritt alle wieder gelöscht.

Auf diese Weise wird durch die Leuchtdioden ein Sägezahnfunktion simuliert, wie sie in der Technik recht häufig vorkommt.

Zur Funtionsweise des Programms:

Immer wenn der Akku mit einem neuen Wert geladen worden ist, gibt er diesen auf die LED's aus. Anschließend wird eine Warteschleife aufgerufen um den Wert eine bestimmte Zeit anzuzeigen.

Nach dieser Wartezeit wird ein neuer, incrementierter Wert in den Akku geladen. Nun werden Sie wieder ausgegeben und der Ablauf geht immer so weiter.

```
                        TITLE   SÄGEZAHN - FUNKTION

                clear   equ     0033h

                        org     1360h

                start:
1360  CD 33 00              call    clear           ;Anzeige löschen

                leucht:                         ;Hauptprogramm
1363  3E FF                ld      a,11111111b     ;Akku mit ffh laden
1365  D3 02                out     (02H),a         ;ausgeben auf die LED's
1367  CD A4 13              call    wait            ;Warteschleife aufrufen

136A  3E FE                ld      a,11111110b     ;eine Leuchtdiode nach der anderen
136C  D3 02                out     (02h),a         ;einschalten und auf die LED's ausgeben
136E  CD A4 13             call    wait            ;Warteschleife aufrufen

1371  3E FC                ld      a,11111100b
1373  D3 02                out     (02h),a
1375  CD A4 13             call    wait

1378  3E F8                ld      a,11111000b
137A  D3 02                out     (02h),a
137C  CD A4 13             call    wait

137F  3E F0                ld      a,11110000b
1381  D3 02                out     (02h),a
1383  CD A4 13             call    wait

1386  3E E0                ld      a,11100000b
1388  D3 02                out     (02h),a
138A  CD A4 13             call    wait

138D  3E C0                ld      a,11000000b
138F  D3 02                out     (02h),a
1391  CD A4 13             call    wait

1394  3E 80                ld      a,10000000b
1396  D3 02                out     (02h),a
1398  CD A4 13             call    wait

139B  3E 00                ld      a,00000000b
139D  D3 02                out     (02h),a
139F  CD A4 13             call    wait

13A2  18 BF                jr      leucht          ;unendlich oft wiederholen

                wait:
13A4  21 00 00             ld      hl,0000h        ;HL-Register mit 0000h laden
13A7  11 01 00             ld      de,0001h        ;DE-Register mit 0001h laden

                loop:
13AA  19                   add     hl,de           ;hl und de addieren
13AB  30 FD                jr      nc,loop         ;bis hl voll (0ffffh)

13AD  C9                   ret                     ;dann zurück ins Hauptprogramm
```


BLINKLICHT

Mit diesem Programm wird ein Blinklicht realisiert, das über die Leuchtdioden angezeigt wird. Die LED's werden immer für eine bestimmte Zeit eingeschaltet und anschließend für die selbe Zeit wieder aus.

Die Funktionsweise des Programms ist recht einfach.

Der Akku wird zunächst mit dem Wert 1111 1111 b geladen. Dies entspricht einem Ausschalten der Leuchtdioden. Dann springt das Programm in eine Wrateschleife. Nach einer dort eingestellten Zeit, kehrt es wieder zurück ins Hauptprogramm. Ein neuer Wert, nämlich 0000 0000 b wird in den Akku geladen. Wieder ein Sprung in die Warteschleife und dann immer weiter so, bis der "RESET"-Taster gedrückt wird.

```
                        TITLE   BLINKLICHT

                clear   equ     0033h

                        org     13B0h

                start:
13B0  CD 33 00          call    clear           ;Anzeige löschen

              blinken:
13B3  3E FF             ld      a,11111111b     ;LED's ausschalten
13B5  D3 02             out     (02h),a
13B7  CD C4 13          call    wait            ;Warteschleife aufrufen

13BA  3E 00             ld      a,00000000b     ;LED7S einschalten
13BC  D3 02             out     (02h),a
13BE  CD C4 13          call    wait            ;Warteschleife aufrufen

13C1  C3 B3 13          jp      blinken         ;unendlich oft wiederholen

                wait:
13C4  21 00 00          ld      hl,0000h        ;hl-Register mit 0000h laden
13C7  11 02 00          ld      de,0002h        ;de-Register mit 0001h laden

                loop:
13CA  19                add     hl,de           ;hl und de addieren
13CB  30 FD             jr      nc,loop         ;bis hl voll (0ffffh)

13CD  C9                ret
```

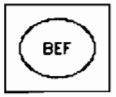

Was bedeuten die Tasten?

Um die nachfolgend beschriebenen Funktionen beobachten zu können, muß Ihr System sich im Befehlsmodus befinden, d.h. Sie müssen vor jedem neuen Test die BEF-Taste bzw. den RESET-Taster drücken.

Für die ersten Tests genügt es, wenn Sie die nachfolgend beschriebenen Tasten kennen.

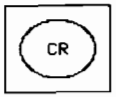
BEF

Bef ist die Abkürzung für Befehl. Mit dieser Taste kann man von den unterschiedlichen Eingabeformen aus, wieder zur Befehlseingabe gelangen. Ihr System befindet sich, nach einem Druck auf diese Taste im selben Zustand wie nach einem RESET. D.h. erwartet einen Befehl wie z.B. START oder STEP etc.



CR

Abkürzung für "Carriage Return" oder auf deutsch für Wagenrücklauf. Diese Funktion kennen Sie bestimmt von einer Schreibmaschine her. Mit dieser Taste werden sämtlichen Eingaben bestätigt. Egal ob Sie einen Befehl eingegeben haben oder ein beliebiges Byte eingetippt haben, müssen Sie die CR-Taste drücken damit Ihr Computer sich den eingegebenen Wert merkt bzw. die angezeigte Startadresse erkennt.



START

Wenn Sie diese Taste drücken, können Sie das an der angezeigten Stelle befindliche Programm starten. Es wird vom System aus immer die Adresse 8100 vorgeschlagen. Wenn Sie mit dieser Adresse einverstanden sind, drücken Sie jetzt die CR-Taste. Wenn nicht geben Sie mit dem Tastenfeld die Adresse ein, an der Ihr Programm sich befindet. Nach dem bestätigen der Adresse startet Ihr Programm.

Beispiel:

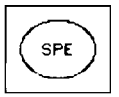
Starten des vorher bereits gestarteten Reaktionstestprogramms:

Eingabe	Anzeige	Bemerkung
BEF	- - - - - - - -	Befehlsmodus einstellen
START	A d r 8.1.0.0	Start
1	A d r 1.0.0.1	
2	A d r 0.0.1.2	Startadresse
3	A d r 0.1.2.3	eingeben
0	A d r 1.2.3.0	
CR		Startadresse bestätigen

Und schon läuft das Programm!

SPE

Abkürzung für Speicher. Nach dem Betätigen dieser Taste können Sie die Inhalte der Speicher anschauen und verändern. Auch hier wird zunächst die Adresse 8100 angezeigt. Wenn Sie damit einverstanden sind, quittieren Sie die Adresse mit der CR-Taste, wenn nicht geben Sie mit der Tastatur eine neue Adresse ein und bestätigen Sie diese mit der CR-Taste. Mit den Tasten +,-und CR kann man durch den Speicher rollen, d.h. man kann sich eine Adresse nach der anderen anschauen und bei Bedarf verändern.



STEP

Wenn Sie diese Taste gedrückt haben läuft das ausgewählte Programm in Einzelschritten ab. Die Funktion ist hauptsächlich zum Testen von Programmen gedacht. Wie bei der Funktion START wird auch hier eine Anfangsadresse angezeigt, im Normalfall 8100. Wenn Ihr Programm an dieser Stelle steht quittieren Sie die Adresse mit der CR-Taste, ansonsten geben Sie eine neue Adresse ein und bestätigen Sie diese mit **CR**. Jetzt haben Sie die Möglichkeit durch das Programm zu rollen, d.h. Sie können nach jedem Druck auf die **CR**-Taste, das erste Byte des vom Programm angesprungenen Befehls anschauen. Ferner haben Sie die Möglichkeit nach einem Druck auf die **BEF**-Taste und einem anschließenden Druck auf die **4 REG**-Taste die einzelnen Registerinhalte

anzuschauen. Wie Sie hierbei vorgehen müssen lesen Sie bitte bei der Beschreibung der **REG**-Taste nach.

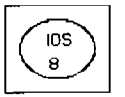
Auch hier ein Beispiel:

Eingabe	Anzeige	Bemerkung
BEF	- - - - - - - -	Befehlmodus einstellen
STEP	A d r 8.1.0.0	Einzelschrittfunktion
1	A d r 1.0.0.1	
3	A d r 0.0.1.3	Startadresse
B	A d r 0.1.3.B	eingeben
0	A d r 1.3.B.0	
CR		Startadresse bestätigen

Mit jedem neuen Druck auf die CR-Taste erscheint der nächste Befehl.

8 IOS

Abkürzung für IO-setzen oder zu deutsch Ein- oder Ausgabeadresse setzen. Mit dieser Funktion ist es möglich Werte auf bestimmte Portadressen auszugeben. D.h. man kann Bitkombinationen auf einer Anzeige ausgeben. Um die Funktion zu testen gehen Sie wie folgt vor. Drücken Sie die **8 IOS**-Taste. In der Anzeige steht jetzt **00** . Der Computer erwartet nun eine Portadresse auf der er die Bytes ausgeben soll. Geben Sie hier z.B. **02** ein und drücken Sie die **CR**-Taste. In der Anzeige erscheint jetzt **02 00**. Jetzt können Sie beliebige Bitkombinationen auf die Leuchdioden ausgeben und die eingegebenen Bytes erscheinen dort als leuchtende und dunkle LED's. Die Ausgabe erfolgt invertiert, d.h. die LED's die hell sind, sind von der Bitkombination her auf Null gelegt. Wenn Sie die Portadresse **00**oder **01** eingeben werden die Bytes auf der Anzeige angezeigt. Hierbei sehen Sie dann rechts nur die eingegebenen Werte.



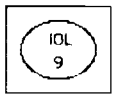
Hierzu ein Beispiel:

Eingabe	Anzeige	Bemerkung
8 IOS	0 0	Funktion anwählen
0	0 0	Portadresse anwählen
2	0 2	
CR	0 2 0 0	Portadresse bestätigen
A	0 2 0 A	Byte eingeben
A	0 2 A A	
CR		

Jetzt leuchtet jede zweite Leuchtdiode auf der HEXIO2

9 IOL

Abkürzung für IO-lesen oder auf deutsch Ein- und Ausgabeadressen lesen. Wenn Sie diese Taste drücken können Sie Bitkombinationen von einem Eingabeport der HEXIO2 einlesen. Gehen Sie wie folgt vor: Zunächst erscheint folgendes Bild in der Anzeige **00** . Wenn Sie jetzt **02** eingeben und **CR** drücken, können Sie die auf dem DIL-Schalter eingestellten Werte als Zahlenkombinationen anzeigen. Nachdem Sie die **CR**-Taste gedrückt haben, erscheint folgendes Bild 02 **XX**. Wobei XX dem eingestellten Wert entspricht. Wenn Sie jetzt die **OPT**-Taste drücken erscheint der angezeigte Wert in binärer Darstellung. Nochmaliges drücken auf **OPT** und er erscheint wieder in hexadezimaler Form.



Und wieder ein Beispiel:

Eingabe	Anzeige	Bemerkung
9 IOL	0 0	Funktion anwählen
0	0 0	Eingabeport anwählen
2	0 2	
CR		Eingabeport bestätigen

Jetzt wird die, im Augenblick auf dem DIL-Schalter eingestellte Bitkombination angezeigt.

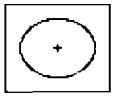
OPT	X X X X X X X X	binäre Ausgabe

Nun steht die eingestellte Bitkombination als Binärzahl in der Anzeige.

Die anschließend besprochenen Funktionen müssen Sie erst beherrschen, wenn Sie weitere Versuche anstellen wollen, oder wenn Sie selbst Programme schreiben wollen.

+

Diese Taste dient zum Anwählen der nächsten Adresse bzw. des nächsten Registers. Sie wird z. B. verwendet um die Speicherinhalte von mehreren Adressen anzusehen. Die Adresse wird jedesmal um eine Ziffer erhöht. In Verbindung mit der **SPE**-Taste kann mit dieser Funktion das komplette Programm Byte für Byte angeschaut werden.



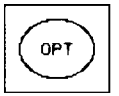
Beispiel:

Eingabe	Anzeige	Bemerkung
SPE	A d r 8.1.0.0.	Speicher anschauen
+	8 1 0 0 X X	Startadresse anwählen
+	8 1 0 1 X X	nächste Adresse anschauen

Auf diese Weise können Sie den kompletten Speicher Ihres Systems anschauen.

OPT

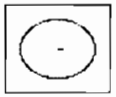
Abkürzung für Option. Dient dem Umschalten, z.B. von Zahlenbereichen oder von Darstellungsarten auf der Anzeige. In Verbindung mit der **UMW**-Funktion können Sie z.B. vom Heaxdezimalen Zahlensystem in das Dezimalsystem umzuwandeln und umgekehrt. Genauere Beschreibung der UMW-Funktion lesen Sie bitte dort nach.



Ein Beispiel hierfür, haben Sie bereits bei der Funktion **9 IOL** kennengelernt.

-

Diese Taste hat die selbe Funktion wie die +-Taste nur, daß damit die Adressen nicht erhöht, sondern decrementiert werden. Nach dem Anwählen der **SPE**-Taste können Sie rückwärts durch die Speicherinhalte rollen.



0 MVE

Abkürzung für move oder auf deutsch bewegen. Mit dieser Taste ist es möglich, einen kompletten Speicherbereich zu verschieben. Der Computer überprüft selbstständig, ob die zu verschiebenden Speicherbereiche sich überlappen. Auf diese Art wird garantiert, daß der Ganze zu übertragende Bereich verarbeitet wird. Um die Funktion auszutesten gehen Sie wie folgt vor: Drücken Sie die **0 MVE**-Taste, in der Anzeige steht jetzt **VON 0000**. Geben Sie die Anfangsadresse der zu verschiebenden Programm stelle ein und drücken Sie die **CR**-Taste. Jetzt steht in der Anzeige **bIS XXXX**. Beben Sie die Endadresse des zu verschiebenden Speicherbereiches ein und quittieren Sie diese. Jetzt steht auf der Anzeige **NAC XXXX**. Geben Sie hier die Adresse ein an die der Speicherbereich verschoben werden soll und quittieren Sie diese Adresse.

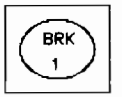
Beispiel:

Eingabe	Anzeige	Bemerkung
0 MVE	V O N 0 0 0 0	Funktion anwählen
8	V O N 0 0 0 8	Anfangsadresse eingeben
1	V O N 0 0 8 1	
0	V O N 0 8 1 0	
0	V O N 8 1 0 0	
CR	b I S 0 0 0 0	Anfangsadresse bestätigen
8	b I S 0 0 0 8	Endadresse festlegen
1	b I S 0 0 8 1	
F	b I S 0 8 1 F	
F	b I S 8 1 F F	
CR	N A C 0 0 0 0	Endadresse bestätigen
8	N A C 0 0 0 8	Zieladresse eingeben
2	N A C 0 0 8 2	
0	N A C 0 8 2 0	
0	N A C 8 2 0 0	
CR	- b E F -	Zieladresse bestätigen

Damit haben Sie den kompletten Speicherbereich übertragen.

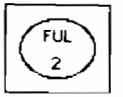
1 BRK

Abkürzung für "break" oder zu deutsch unterbrechen. Es lassen sich mit dieser Taste bis zu drei Unterbrechungen setzen, an denen das Programm dann automatisch unterbrochen wird. Zum Testen von Programmen ist das sehr nützlich. Vorgehensweise wie folgt: Nach dem Drücken der **1 BRK**-Taste erscheint auf der Anzeige **01 0000**. Hier können Sie den ersten **Break** setzen. Nach dem Bestätigen dieser Adresse können Sie noch zwei weitere Breaks auf die gleiche Art setzen. Beim Ablaufenlassen des Programms hält es zunächst an der von Ihnen als Breakadresse gesetzten Speicherstelle anhalten. Wenn Sie Ihr Programm dann von dieser Adresse neu starten bleibt es bei dem nächsten gesetzten Break stehen usw. Wenn Sie nur einen Break setzen wollen geben Sie nur für diesen eine Adresse ein, die anderen beiden können Sie dann durch einfaches Quittieren der angezeigten Adresse(**02 0000**) bzw. (**03 0000**) ungültig machen. Nach jedem Break können z.B. die Register angeschaut werden.



2 FUL

Abkürzung für füllen. Mit dieser Taste kann man einen Speicherbereich mit einem konstanten Wert auffüllen. Der Rechner überprüft automatisch ob alle angesprochenen Adressen auch richtig belegt wurden. Betätigen Sie die **2 FUL**-Taste. Es erscheint **VON 0000** auf der Anzeige. Geben Sie die Anfangsadresse des zu füllenden Bereiches an und quittieren Sie diese Adresse. Nun erscheint **bIS 0000** in der Anzeige. Geben Sie hier die Endadresse des zu füllenden Speicherbereiches ein (quittieren nicht vergessen). Jetzt müssen Sie nur noch einen Wert eingegeben mit dem der Speicherbereich gefüllt wird. In der Anzeige steht jetzt **dtA 00**. Nach der Eingabe erscheint wieder die **-bEF-** -Meldung.

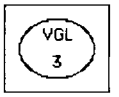


Noch ein Beispiel:

Eingabe	Anzeige	Bemerkung
2 FUL	V O N 0 0 0 0	Funktion anwählen
8	V O N 0 0 0 8	Anfangsadresse eingeben
9	V O N 0 0 8 9	
0	V O N 0 8 9 0	
0	V O N 8 9 0 0	
CR	b I S 0 0 0 0	Anfangsadresse bestätigen
8	b I S 0 0 0 8	Endadresse eintippen
9	b I S 0 0 8 9	
F	b I S 0 8 9 F	
F	b I S 8 9 F F	
CR	d t A 0 0	Endadresse bestätigen
A	d t A 0 A	Datenwert eingeben
B	d t A A b	
CR	- b E F -	Datenwert bestätigen

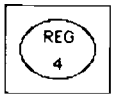
Jetzt steht im von Ihnen eingetippten Speicherbereich nur noch der Wert **AB**.

3 VGL



Abkürzung für Vergleich. Zwei Speicherbereiche werden miteinander verglichen. In der Anzeige erscheinen die selben Abfragen wie bei der **0 MVE**-Funktion. Sie müssen jetzt die zu vergleichenden Speicherbereiche eingeben. Treten irgendwelche Unterschiede beim Vergleich auf wird die Meldung **PRF XXXX** ausgegeben, wobei XXXX für die fehlerhafte Adresse steht. Ist der Vergleich in Ordnung erscheint die **-bEF-** ausgegeben.

4 REG



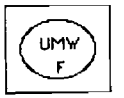
Abkürzung für Register. Nach dem Drücken auf diese Taste werden die Inhalte der Register des Computers angezeigt. Jetzt können die Registerinhalte angeschaut und verändert werden. In der Anzeige erscheint folgendes:
AF XXXX wobei die ersten beiden Zahlen den Akkuinhalt anzeigen und die rechten beiden den Inhalt des Flagregisters. Jetzt können Sie die die Inhalte der beiden Register verändern. Wenn Sie nur den Inhalt des Akkus verändern wollen, müssen Sie anschließend das Byte, das im

Flag stand, neu eingeben. Es können aber auch die anderen Register angeschaut bzw. verändert werden. Dazu müssen Sie die Tasten + oder **CR** drücken. Dann erscheinen nacheinander die anderen Register des Computers. Mit der - Taste gehen Sie in die andere Richtung. Es sind der Reihe nach:

AF XXXX
BC XXXX
DE XXXX
HL XXXX
SP XXXX
IX XXXX
IY XXXX
AF' XXXX
BC' XXXX
DE' XXXX
HL' XXXX
rI' XXXX

XXXX steht jeweils für den Inhalt des ausgewählten Registerpaares.

F UMW



Abkürzung für Umwandlung. Mit dieser Funktion können Sie dezimal vorgegebene Zahlen in hexadezimale Zahlen umrechnen. Das ist gerade beim Einstieg in die Programmierung recht hilfreich. Um Zahlensysteme ineinander umzuwandeln müssen Sie folgende Eingaben machen: Nach dem aktivieren der Funktion erscheint in der Anzeige **0000**. Jetzt können Sie eine hexadezimale Zahl eingeben. Nach dem Druck auf die **CR**-Taste erscheint der dezimale Wert dieser Zahl. Nochmaliges Betätigen der **CR**-Taste bringt Sie zurück zur Eingabe einer neue Zahl. Wenn Sie aber die **OPT**-Taste drücken erscheint die Zahl auch in der dezimalen Darstellung. Nochmals **OPT** und es steht wieder die hexadezimale Darstellung in der Anzeige. Wenn nach der **OPT** Betätigung die **CR**-Taste gedrückt wird kann eine dezimale Zahl eingegeben werden. Sie können auch Dezimalzahlen mit Vorzeichen eingeben nur können Sie nach der Vorzeicheneingabe keine weiteren Ziffern Ihrer Zahl eintippen. Die Umwandlung erfolgt wie oben.

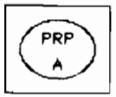
Auch hier noch ein Beispiel:

Eingabe	Anzeige	Bemerkung
F UMW	0 0 0 0	Funktion aufrufen
Eingabe einer beliebigen Zahl; wie vorher beschrieben		
CR	d X X X X X	Eingabe Bestätigung
CR	0 0 0 0	
neue Zahl eingeben		
OPT	d X X X X X	dezimal anzeigen
OPT	S X X X X	hexadezimal anzeigen
CR	0	umstellen auf dezimale Eingabe

Jetzt können Sie dezimale Zahlen eintippen und diese dann hexadezimale Zahlen umwandeln.

Die Funktionen die jetzt noch beschrieben werden, sind nur einzusetzen wenn Sie eine zusätzliche Steckerleiste in den zweiten Steckplatz einlöten. Dann könne Sie noch eine zusätzliche Baugruppe aufstecken. Das kann z.B.eine CAS-Baugruppe oder eine Prommer-Baugruppe sein.

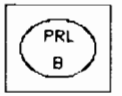
A PRP



Abkürzung für Eprom-Programmierung. Ein Eprom ist ein Bauteil, mit dem Sie Ihre Programme in den Festwertspeicher eines Rechners schreiben können. Mit dieser Taste ist es nun möglich Programme auf Eproms zu speichern. Ein Eprom (Erasable Programmable Read Only Memory) ist ein Baustein der frei programmierbar ist, aber nur noch mit einem speziellen UV-Löschgerät gelöscht werden kann. Diese Eproms können dann in einen Rechner eingesetzt werden und sind dann dort wie ein Festwertspeicher einsetzbar. Um ein EPROM zu programmieren, benötigen Sie die **Prommer**-Baugruppe. Näheres dazu finden Sie im Prommer-Handbuch. Es werden wie bei den **LAD** und **SPE** Funktionen die selben Eingaben verlangt. Ist der Programmiervorgang fehlerhaft erscheint auch hier die **PRF XXXX**-Meldung. Ist er aber richtig abgeschlossen erscheint die

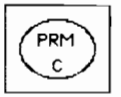
Meldung **burnEd** in der Anzeige, was soviel wie gebrannt bedeutet.

B PRL



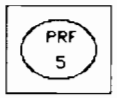
Abkürzung für Eprom-Lesen. Damit ist es möglich einen Eprom-Inhalt in einen Speicherbereich Ihres Systems zu übertragen. Das System verlangt nach dem Aufrufen der Funktion zunächst die Anfangsadresse des zu übertragenden Speicherbereiches, mit der Meldung **VON** 0000. Hier wird nomalerweise der Wert 0000 bestätigt. Jetzt verlangt das System die Größe des zu übertragenden Bereiches in dem es die Endadresse abfragt.(**bIS 07FF**) Wenn Sie den ganzen Speicherbereich des EPROMs übertragen wollen quittieren Sie mit der CR-Taste. Nun muß nur noch die Anfangsadresse im Speicher eingegeben werden (**NAC 8100**). Sind Sie mit der angezeigten Adresse 8100 einverstanden, drücken Sie die CR-Taste, ansonsten geben Sie einen anderen Wert ein. Tritt beim Laden ein Fehler auf z.B. kein freier Speicherbereich bei der angegebenen Adresse so erscheint die **PRF XXXX**-Meldung. XXXX ist die Adresse bei der der Fehler auftritt.

C PRM



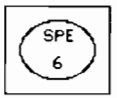
Abkürzung für Prommer. Diese Taste dient dazu, eine angeschlossene Prommerbaugruppe an Ihr System anzupassen. Ist eine Prommerbaugruppe angeschlossen und wird diese Taste gedückt, erscheint die Monoflop-Zeitdauer in Millisekunden auf der Anzeige. Diese müssen Sie auf **50 ms** einstellen. Die Einstellung erfolgt mit dem Trimmer, der auf der Prommer-Karte angebracht ist. Es ist nicht unbedingt nötig den Wert bis zur letzten Ziffer einzustellen.

5 PRF



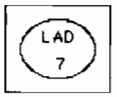
Abkürzung für Prüfen. Daten werden vom Kassettenrekorder mit dem aktuellen Speicherinhalt verglichen. Hierfür benötigen Sie die **CAS**-Baugruppe, die Sie auf die HEXIO2 oder auf einen angeschlossenen Bus stecken müssen. Ist der Vergleich zwischen Speicherinhalt und Kassetteninhalt positiv wird die Meldung **St= XXXX** ausgegeben. Wenn irgendein Fehler auftritt wird erscheint **PRF XXXX** in der Anzeige. XXXX steht für die Adresse an der der Fehler auftrat. Eine weitere Fehlermöglichkeit ist ein Prüfsummenfehler der durch die Meldung **CHE** gemeldet wird. Ist der Vergleich fehlerlos, erscheint die **-bEF-**Meldung in der Anzeige.

6 SPE



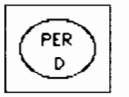
Abkürzung für Speichern. Die Programme des Arbeitsspeichers werden auf Kassette gespeichert. In der Anzeige erscheint zunächst eine **VON XXXX** Anfrage. Hier geben Sie die Anfangsadresse des zu übertragenden Speicherbereiches ein. Nach einem Druck auf die **CR**-Taste erscheint die Meldung **bIS XXXX** in der Anzeige. Hier geben Sie die Endadresse des zu übertragenden Bereiches ein. Bevor Sie die **CR**-Taste drücken müssen Sie aber den Kassettenrecorder einschalten. Der Übertrag wird mit einer Prüfsumme übertragen, so daß beim laden von Kassette eine zusätzliche Kontrolle gegeben ist.

7 LAD



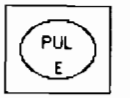
Abkürzung für laden. Mit dieser Funktion können Daten von einem Kassettenrekorder eingelesen werden. Die CAS-Baugruppe muß angeschlossen sein. Eine Kontrolle der übertragenen Daten wird genauso, wie bei der **6 SPE**-Funktion durchgeführt. Auch die Fehlermeldungen sind die selben. **PRF XXXX** für eine fehlerhafte Übertragung und **CHE** für einen Prüfsummenfehler. Bei einem richtigen Datentransfer erscheint in der Anzeige die Meldung **St= XXXX**, wobei XXXX die Startadresse des Programms ist. Ein weiterer Tastendruck und die **-bEF-** -Meldung erscheint. Jetzt kann normal weitergearbeitet werden. Genauere Angaben zur CAS-Baugruppe entnehmen Sie bitte dem entsprechenden Handbuch.

D PER



Abkürzung für Periodendauer. Mit dieser Funktion können Sie eine Periodendauer messen. Um die Funktion zu verwenden müssen Sie die CAS-Baugruppe einstecken und eine zusätzliche Leitung legen. Bit 7 des I/O-Ports 0 wird mit einer Leitung verbunden. Dies ist die Meßleitung. Diese Leitung wird mit Pin4 des 6850 auf der CAS-Baugruppe verbunden (Lötseite).Sind diese Veränderungen durchgeführt, erscheint die Periodendauer in ys auf der Anzeige. Jetzt können Sie mit dem Trl (Trimmer1) die Periodendauer auf ca. 833ys abgleichen. Siehe auch CAS-Handbuch.

E PUL



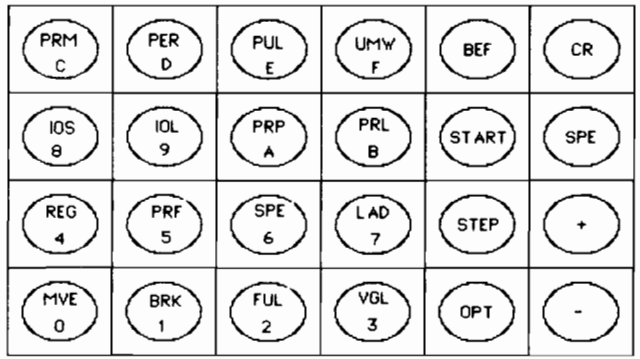
Abkürzung für Pulsdauer. Diese Taste hat eine ähnliche Funktion wie die vorher beschriebene Taste. Der Unterschied zur vorherigen Funktion besteht darin, daß hier die 0-Dauer des Signals gemessen wird. Dazu wird das vorher angebrachte Meßkabel mit dem Pin3 des 6850 auf der CAS-Baugruppe verbunden. Mit dem Trimmer 2 (Tr2) kann dann der Wert auf ca. 625ys eigestellt werden. Dazu muß die CAS-Baugruppe auf Aufnahme gesteckt werden und der Teststecker eigesteckt werden. Siehe auch CAS-Handbuch.

Jetzt nochmal in Kurzform

Taste	Bedeutung und Funktion
BEF	Befehlmodus einstellen
CR	Carriage Return, Bestätigung aller Eingaben
START	starten eines Programms
SPE	Speicherbereich anschauen, bzw. verändern
STEP	Einzelschrittfunktion, Programm kann in Einzelschritten ablaufen gelassen werden
+	nächste Stelle aufrufen, z.B. Speicherstellen oder Registerinhalte
OPT	arbeitet nur im Zusammenhang mit anderen Funktionen, dient dem Umschalten von Zahlenebenen
-	gleiche Funktion wie + , nur jeweils die vorherige Speicherstelle wird aufgerufen
0 MVE	Speicherbereiche verschieben, ein Speicherbereich wird in einen anderen verschoben
1 BRK	Breakpunkte setzen, Unterbrechungen für Programme können gesetzt werden, um z.B. Registerinhalte anzuschauen
2 FUL	Speicherbereiche mit einem frei wählbaren Wert füllen
3 VGL	zwei Speicherbereiche werden miteinander verglichen
4 REG	Registerinhalte der einzelnen Register anschauen mit + , - oder CR kann man alle Register nacheinander anschauen
5 PRF	prüft die Daten die auf Kassette geschrieben wurden, indem er sie mit dem entsprechenden Speicherbereich vergleicht, CAS-Baugruppe muß aufgesteckt werden
6 SPE	speichert Daten auf Kassette, bildet gleichzeitig eine Prüfsumme damit beim späteren Lesen eine Kontrolle der übertragenen Daten möglich ist CAS-Baugruppe muß aufgesteckt werden
7 LAD	Programme von Kassette laden, abgespeicherte Prüfsumme wird mit eingelesener verglichen CAS-Baugruppe muß aufgesteckt werden
8 IOS	Ausgabeport festlegen und Bytes auf diesen ausgeben

Taste	Bedeutung
9 IOL	Eingabeport festlegen und dort eingestellte Bytes auf der Anzeige ausgeben, mit OPT eine Ausgabe in binärer und hexadezimaler Form möglich
A PRP	Eproms programmieren, ein Speicherbereich kann auf ein Eprom programmiert werden, PROMMER-Baugruppe muß verwendet werden
B PRL	übertragen eines Eprominhalts in einen Speicherbereich, PROMMER-Baugruppe muß verwendet werden
C PRM	Funktion zum abgleichen der PROMMER-Baugruppe, auf der Anzeige erscheint ein Zahlenwert in ms, für PROMMER muß dieser Wert auf 50ms abgeglichen werden
D PER	Periodendauer einer anliegenden Frequenz wird gemessen, Meßleitung muß angeschlossen werden, dann wird kann die Periodendauer in ys gemessen werden, für die CAS-Baugruppe muß sie auf 833ys eingestellt werden
E PUL	Funktion mißt die Nulldauer eines Signals, für CAS-Baugruppe muß sie 624ys betragen
F UMW	hexadezimal in dezimal umwandeln und umgekehrt, mit OPT und CR kann umgeschaltet werden

TASTATUR HEXI02



Stichwortverzeichnis

Byte:

Ein Byte ist die gebräuchliche Bezeichnung für eine Dateneinheit. Zusammengesetzt wird ein Byte, aus acht Bits. Ein Bit ist ein Wert der entweder 0, oder 1 sein kann.

Listing:

Die Liste der, in einem Programm verwendeten Befehle.

RAM:

Random-Access-Memory ist der Arbeitsspeicher Ihres Systems. Im RAM könne sie schreiben, d.h. selbst Daten und Programme eingeben und lesen. Er beginnt beim Einsteigerpaket auf 8100. Dies ist vom Grundprogramm her so festgelegt worden. Deshalb erscheint nach der Betätigung einer Funktionstaste immer die Anfangsadresse 8100.

ROM:

Read-Only-Memory ist der Festwertspeicher Ihres Systems. Er besteht aus dem Monitorprogramm und den anderen Programmen die in Ihrem System fest installiert sind. Das Monitorprogramm sorgt dafür, daß eine Verbindung "Mensch-Maschine" möglich ist.